論文名稱：應用深度學習於時空資料預測　　頁數：71

校系（所）組別：淡江大學 資訊工程學系 碩士班

畢業時間及提要別：111 學年度第一學期 碩士論文提要

研究生：郭有富　　指導教授：蔡憶佳 博士

論文提要內容：

在本論文中，對時空數據挖掘網絡進行了廣泛的探討並使用火災事件數據集對這些網絡模型進行了比較。本文解決兩個問題： 1. 在最近提出的 STDM-DL（時空數據挖掘，深度學習）模型中，比較這些模型的預測能力？ 2. 當應用於火災數據集時，這些模型的性能如何？本論文進行了兩個實驗。第一個是使用他們的數據運行最先進的 STDM-DL 模型並比較它們的性能。本研究下的模型由 METR-LA 或 PEMS-BAY 數據集訓練，預測空間和時間域中的交通。在第二個實驗中，我們使用了新北市的火警數據集 (NTPC-Fire 2015-17) 並實現了一些熟悉但簡單的模型，例如自動編碼器和 GAN，以重建（預測）光柵化熱圖。然後，我們使用 LSTM-RNN、FBProphet 和 ARIMA 處理時間表示，以比較每日和每週事件頻率的時間序列預測的性能。我們的第一個實驗發現一些最先進的模型，例如 ST-METANET、STGCN 和 Spacetimeformer，都具有相似的性能。Multi-LSTM 優於所有這些模型。 “Deepforecast Multi-LSTM”是迄今為止最好的交通預測模型。令人驚訝的是，在我們的第二個實驗中，對於我們的數據集，FBProphet 模型以 6.97231 RMSE 和 5.045342 MAE 優於最好的 LSTM。同樣，我們重建（預測）柵格熱圖的最佳空間模型是具有 1.04198155 RMSE 和 0.3522904 MAE 的 9 批變分自動編碼器 (VAE)。鑑於這些發現，我們進一步使用數據可視化並為 STDM 任務中的每個域實施組合模型和架構。這項研究表明，這些現有模型可用於解決時空領域的預測問題。

Title of Thesis:	Applications of Deep Learning in GIS – Spatiotemporal Data Mining and Forecasting	Total Pages: 71
Keywords:	Problem Solving, Data Mining, Deep learning, GIS, Geo-Information System, Spatiotemporal Data mining	
Name of Institute:	Master's Program, Department of Computer Science and Information Engineering, Tamkang University	
Graduation date:	2/2022	Degree conferred: Master
Name of Student: Supasin Wuthikulphakdi		Advisor: Dr. Yihjia Tsai

ABSTRACT:

In this thesis, an extensive survey on spatiotemporal data mining networks was done, and a comparison of those network models using a fire event dataset was made. This paper addresses two issues: 1. Among those recently proposed STDM-DL (spatial-temporal data mining, deep learning) models, how to compare those models in forecasting capability? And 2. What are the performances of those models when applied to the fire dataset from our previous search? Two experiments are performed in this paper. The first one was that we ran the state-of-the-art STDM-DL models using their data and compared their performances. The models under this study are trained by either METR-LA or PEMS-BAY dataset, predicting the traffic in spatial and temporal domains. In the second experiment, we used the fire-call dataset of New Taipei (NTPC-Fire 2015-17) and implemented some familiar yet straightforward models, such as Autoencoders and GANs, to reconstruct (predict) rasterised heatmaps. Then, we processed the temporal representations using an LSTM-RNN, FBProphet, and ARIMA, to compare performance in time series forecasting of daily and weekly incident frequencies. Our first experiment discovered that some state-of-the-art models, such as ST-METANET, STGCN, and Spacetimeformer, all have similar performance. The multi-LSTM outperformed all those models. The "Deepforecast Multi-LSTM" is the best traffic prediction model to date. Surprisingly, in our second experiment, for our small dataset, the FBProphet model outperformed our best LSTM with 6.97231 RMSE and 5.045342 MAE. Likewise, our best spatial model to reconstruct (predict) a raster heatmap was the 9-Batch Variational Autoencoder (VAE) with 1.04198155 RMSE and 0.3522904 MAE. Given these findings, we further use data visualisation and implement combined models and architectures for each domain in the STDM task. This study suggested that those existing models can be used to solve issues in the spatial-temporal domain, and the use of deep learning networks is a fast-growing research field that depends intensely on big data.

ACKNOWLEDGEMENTS

I cannot express enough thanks to my committee for their continued support and encouragement: Dr. Yihjia Tsai (蔡憶佳), my committee chair; Dr. Hwei-Jen Lin (林慧珍); Dr. Ching-Chang Lin (林慶昌); and Mr Chien-Tzung Lee (李建宗). I offer my sincere appreciation for the learning opportunities provided by my committee.

My completion of this project could not have been accomplished without the support of my classmates, Zhouwei, Guanren, Xiaoxuan, Ah-Hao, Yating, and Cifen
Thanks to my parents as well, Mr. and Mrs. Wuthikulphakdi (วุฒิกุลภักดี). The countless times you kept the financial and the mental support during our hectic schedules will not be forgotten.

Finally, to my caring, loving, and supportive best friend, Sirichai Sawaengsab (ศิริชัย แสวงทรัพย์): my deepest gratitude. Your encouragement when the times got rough are much appreciated and duly noted. It was a great comfort and relief to know that you were willing to provide care for my old parents while I completed my work. My heartfelt thanks.

Supasin WUTHIKUPHAKDI
January 2022

CONTENTS

List of Figures

List of Tables

CHAPTER1: INTRODUCTION & BACKGROUND

1.1 Overview, Background & Purpose

The goal of this thesis is to study how deep learning architectures and models can be implemented in ST-data mining in traffic flow prediction, intelligent transport system (ITS) and weather/disaster forecast. And to discover what group of models, to date, is the best for STDM, especially in fire incidents forecasting and traffic flow prediction – such as ST-LSTM and LT-LSTM.

As defined in [17], spatiotemporal data is a dataset related to both space and time. It is collected in the space domain (a.k.a. a (heat) map over a location in the form of raster and vector data matrices) and the time domain (a.k.a. the time-series - a heatmap average for each timestep in the form of timestep-D vectors). Overall, when concatenated, a spatiotemporal dataset is usually a 3D+ tensor.

Given the popularity of GPS-supporting devices, including smartphones, the need for S.T.-data mining rises with the emerging use of intelligent transport systems (ITS, i.e., automatic guidance, self-driving cars, traffic prediction, etc.), and disaster prediction (i.e., weather, flood, earthquake, storms, clouds, smog, global warming etc.).

To manually collect, process, and forecast the ST data is a laborious task. Several models based on both machine learning have been deployed, but ML models need a human to extract the feature representations. Deep learning models, capable of self-feature extraction, usually outperform regular ML ones. Fundamentally, an ST-Deep Learning model works as follows:

- A CNN learns the spatial input (i.e., heatmap, graph data (GCN), & data coordinates)
- While a sequential model, like LSTM/RNN, learns the temporal domain of the input data.
- With both DNNs working together, sometimes with an intermediate submodel (i.e., a dense submodel in [14]), we can correlate and forecast spatiotemporal data (such as traffic, train ridership, etc.) not only the time-series (time-domain) but also the heatmap (space domain).

However, DL models have drawbacks; a CNN/GCN/FCN can only extract and learn only the space domain, while an RNN/LSTM/seq2seq can only do these on the temporal domain of the ST-data. Proven by several papers, RNN, when processing a long sequence, has a "gradient vanishing and explosion" problem. Thus, LSTM-based architectures are preferred in later works.

As a result, some works in section 2 overcame the problems by adding such two neural networks into the same framework, allowing them to extract and learn the ST-data of both domains simultaneously. More technical details are further readable in [8] and [19].

The first part of this paper studies deep learning algorithms and models in ST-data mining & forecasting. The following is a shortlist of spatiotemporal data mining models:

- CNN-GCN-FCNs are used to extract features in the spatial domain (i.e., heatmap).
- LSTM-RNN-seq2seq are used to extract features in the temporal domain (i.e., sequence type data).
- In land usage survey, some researchers use CNN to study changes in land use, crop growth, and construction progresses.
- In ITS, ST-Data mining is extremely useful in traffic prediction, guidance systems, route planning, and self-driving cars.

- In Meteorology, the STDM is used for weather and disaster forecast.
- To make GPS based pathfinding more accurate, Spatiotemporal Data Mining is often utilized in several papers to train the pathfinding ML/DL-architectures.
- In transportation, deep learning methods learn highly intricate ST-correlations among the traffic data – useful in some tasks such as traffic flow prediction, traffic incident detection, and traffic congestion prediction – such as in [13] and [14].
- In On-demand services, such as Uber [10], to perform pathfinding
- All the referred papers involve GIS, Computer Vision (CV), time-series prediction, and spatiotemporal data mining.

1.2 Outline & Technical Concepts

According to [8], [17], [19], and [76], there are some fundamentals issues of STDM that need to be addressed before using deep learning networks:

- Spatiotemporal Data Structures
- Data Instances
- Data Representations
- Deep Learning Models in STDM

1.2.1 Data instances.

Usually, a spatiotemporal dataset is a 3D tensor indicating space (like a book page) and time (number of pages). The ST data can generally be summarized into the following data instances: points, trajectories, time-series, spatial maps, and ST raster, as shown in the left part of Figure 4. An ST point can be represented as a tuple containing the spatial and temporal information and some additional observation features, such as the types of crimes or traffic accidents. Besides ST events, trajectories and ST point references can also be formed as points. For example, one can break a trajectory into several discrete points to count how many trajectories have passed a particular region in a particular time slot.

Different data instances can be extracted from ST raster as time-series, spatial maps, or ST raster itself, depending on different applications and analytic requirements:

1. We can consider the measurements at a particular ST grid of the ST field as a time-series for some time-series mining tasks.
2. For each timestamp, an ST raster's measurements can be considered a spatial map.
3. One can also consider all the measurements spanning locations and time stamps.

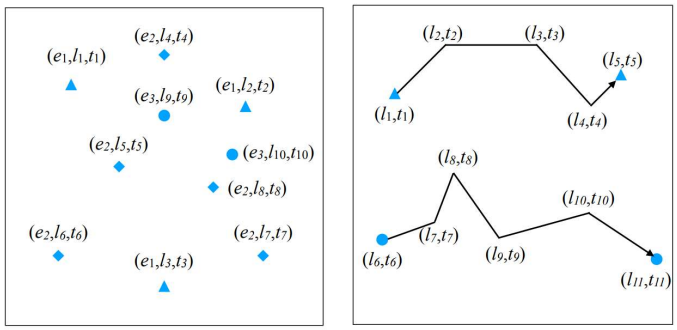
All these considerations are for analysis. The ST raster itself can be a data instance in such a case.

1.2.1.1 Event data.

Event data comprise of discrete events occurring at point locations and times (e.g., crime events in the city and traffic accident events in a transportation network). An event can generally be characterized by a point location and time, which denotes where and when the event occurred, respectively. For example, a crime event can be characterized as such a tuple (e_i, l_i, t_i), where e_i is the crime type, l_i is the location where the crime occurs and t_i is the time when it occurs [8, 19].

1.2.1.2 Trajectory data

Trajectories denote the paths traced by bodies moving in space over time. (e.g., the moving route of a bike trip or taxi trip). Trajectory data are usually collected by the sensors deployed on the moving objects that can periodically transmit the object's location over time, such as GPS on a taxi. Figure 1(b) shows an illustration of two trajectories. Each trajectory can be usually characterized as such a sequence $\{(l_1, t_1), (l_2, t_2) \ldots (l_n, t_n)\}$, where li is the location (e.g., latitude and longitude) and t_i is the time when the moving object passes this location [8].



(a) Three types of events (b) The trajectories of two moving objects

Figure 1: Illustration of event and trajectory data types [8]

1.2.1.3 Point reference data.

Point reference data consist of measurements of a continuous ST field such as temperature, vegetation, or population over a set of moving reference points in space and time. For example, meteorological data such as temperature and humidity are commonly measured using weather balloons floating in space, continuously recording weather observations. Point reference data can be usually represented as a set of tuples as follows $\{(r_1, l_1, t_1), (r_2, l_2, t_2) \ldots (r_n, l_n, t_n)\}$. Each tuple (r_i, l_i, t_i) denotes the measurement of a sensor r_i at the location l_i of the ST filed at time t_i. [8]

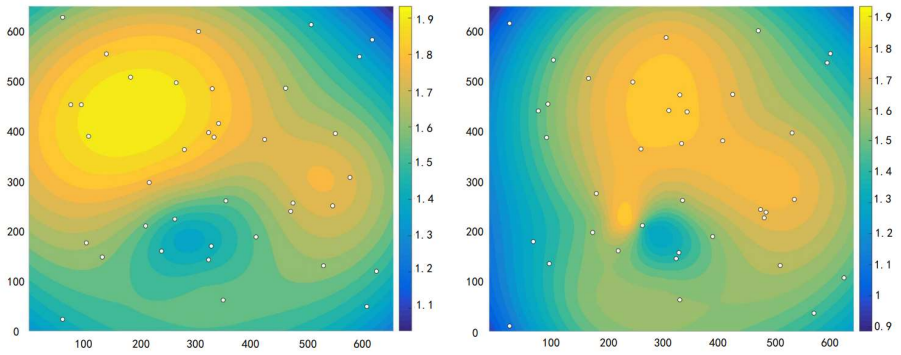


Figure 2: Illustration of ST reference point data in two timestamps. The while circles are the locations of the sensors that record the readings of the ST field. The colour bars show the distribution of the ST field. [8]

1.2.1.4 Raster data.

Raster data are the measurements of a continuous or discrete ST field that are recorded at fixed locations in space and at fixed time points. The major difference between point reference data and raster data is that the locations of the point reference data keep changing while the locations of the raster data are fixed. The locations and times for measuring the ST field can be regularly or irregularly distributed. Given m fixed locations $S = \{s_1, s_2, \ldots s_m\}$ and n time stamps $T = \{t_1, t_2, \ldots t_n\}$, the raster data can be represented as a matrix $R^{m \times n}$, where each entry r_{ij} is the measurement at location s_i at time stamp t_j. Raster data are also quite common in real-world applications such as transportation, climate science, and neuroscience. For example, Figure 3 shows an example of the traffic flow raster data of a transportation network. Each road is deployed a traffic sensor to collect real-time traffic flow data. The traffic flow data of all the road sensors in a whole day (24 hours) form a raster data. [8]

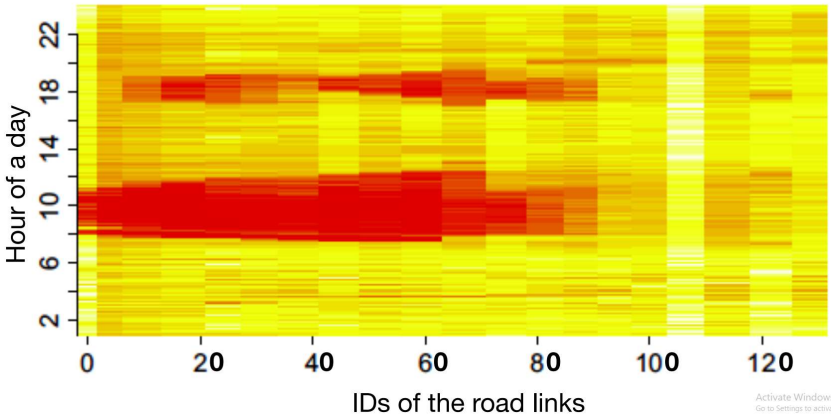


Figure 3: Illustration of raster data collected from traffic flow sensors. The x-axis is the ID of the road links in a transportation network, and the y-axis is the hour of a day. Different colors denote different traffic flows on the road links captured by the road sensors deployed at fixed locations. [8]

1.2.1.5 Video.

A video, consisting of a sequence of images, can also be considered a type of ST data. In the spatial domain, the neighboring pixels usually have similar RGB values and thus present high spatial correlations. In the temporal domain, the images of consecutive frames usually change smoothly and present high temporal dependency. A video can be generally represented as a three-dimensional tensor with one dimension representing time t and the other two representing an image. Furthermore, video data can also be considered special raster data if we assume that there is a "sensor" deployed at each pixel, and the "sensors" will collect the RGB values at each frame. [8]

1.2.2. Data Representations.

For the abovementioned five types of ST data instances, four types of data representations are generally utilized to represent them as the input of various deep learning models, sequence, graph, 2-dimensional matrix, and 3-dimensional tensor, shown in the right part of Figure 4. Different deep learning models require different types of data representations as input. Thus, representing the ST data instances relies on the data mining task under study and the selected deep learning model. [8]

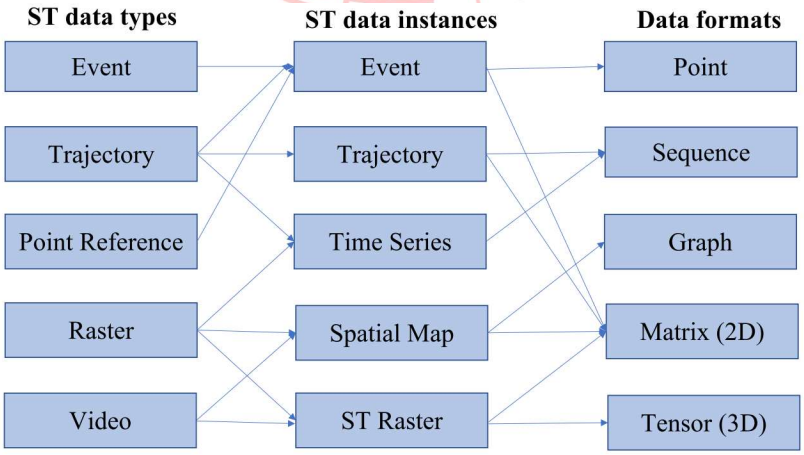


Figure 4: Data instances and representations of different ST data types [8]

1.2.3. Deep learning models.

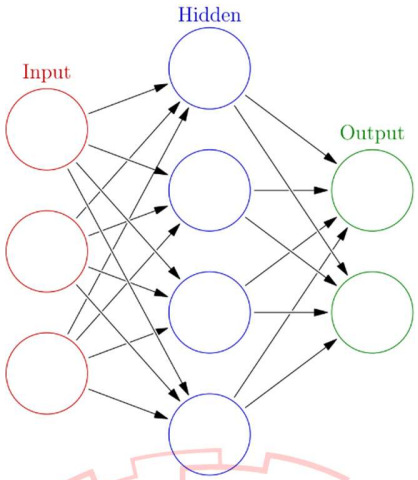


Figure 5: a diagram of an Artificial Neural network [76]

Deep learning models, praised for their outstanding performance and accuracy, are often utilized in several STDM papers. There are some notable examples below.

1.2.3.1 Restricted Boltzmann Machines (RBM).

A Restricted Boltzmann Machine is a two-layer stochastic neural network [53] that can be used for dimensionality reduction, classification, feature learning and collaborative filtering. RBM is usually used for learning features. As shown in Figure 6, the first layer of the RBM is called the visible, or input layer with the neuron nodes $\{v_1, v_2, ...v_n\}$, and the second is the hidden layer with the neuron nodes $\{h_1, h_2, ...h_m\}$. As a fully connected bipartite undirected graph, all nodes in RBM are connected to each other across layers by undirected weight edges $\{w_{11}, ...w_{nm}\}$, but no two nodes of the same layer are linked. The standard type of RBM has a binary-valued node and bias weights. RBM tries to learn a binary code or representation of the input, and depending on the task, RBM can be trained in either supervised or unsupervised ways. [8, 19]

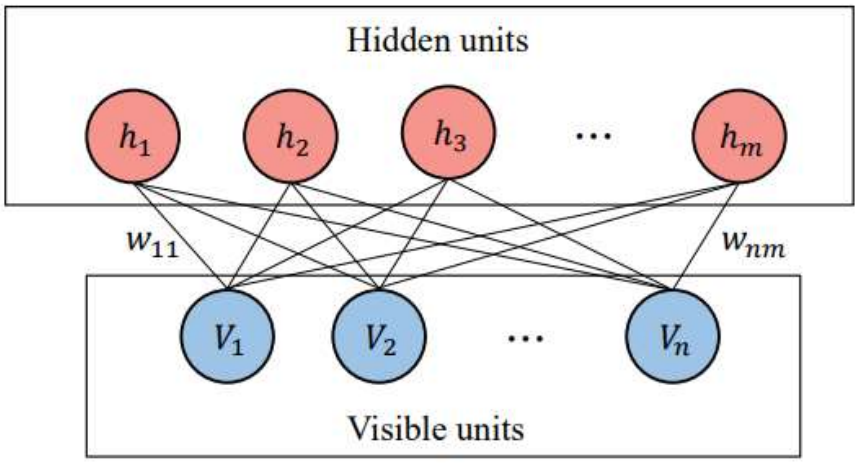


Figure 6: Structure of the RBM model [8]

1.2.3.2 Convolutional Neural Networks (CNN).

Convolutional neural networks (CNN) are a class of deep, feed-forward artificial neural networks applied to analyze visual imagery. A typical CNN model usually contains the following layers as shown in Figure 7: the input layer, the convolutional layer, the pooling layer, the fully connected layer, and the output layer. [8]

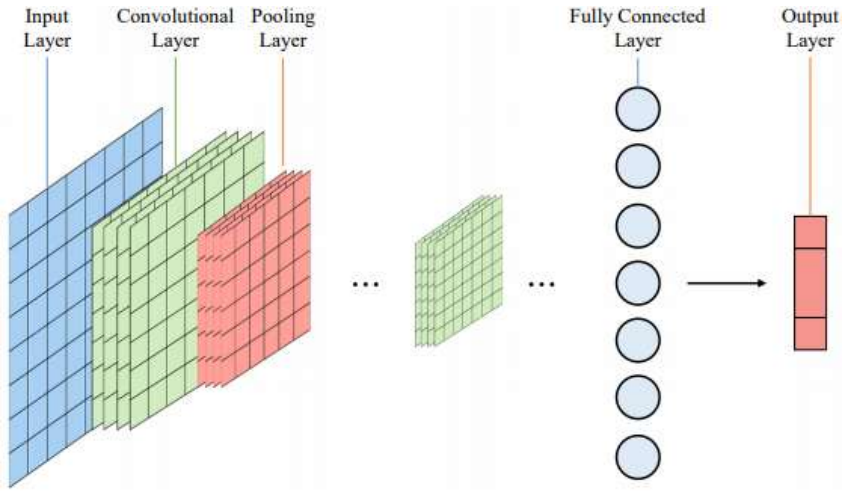


Figure 7: Structure of the CNN model [8]

First, the convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. Second, the pooling layer will then downsample the spatial dimensionality of the given input to reduce the number of parameters. The fully connected layers will connect every neuron in one layer to every other in the next layer to learn the final feature vectors for classification.

The CNN architecture is designed to process image data. Due to its powerful ability to capture spatial domain correlations, it is now widely used in mining ST data, especially spatial maps and ST raster heatmaps. [19]

1.2.3.3. Graph CNN (GCNN).

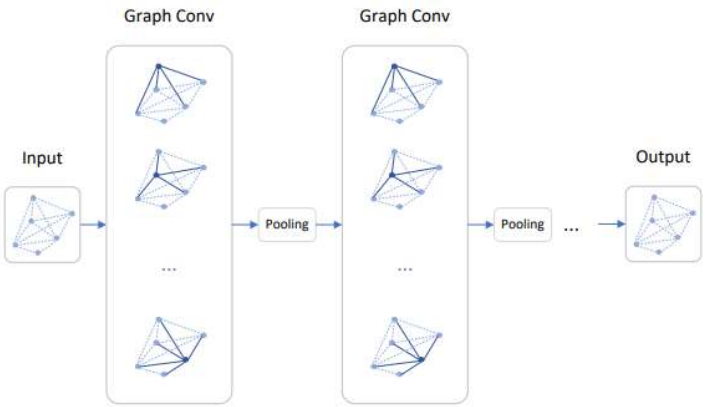


Figure 8: Structure of Graph CNN model [8]

CNN is designed to process images that can be represented as a regular grid in the Euclidean space. Graph CNN has recently been widely studied to generalize CNN to graph-structured data [160]. Figure 8 shows a structure illustration of a Graph CNN model. The graph convolution operation applies the convolutional transformation to the neighbours of each node, followed by the pooling operation. By stacking multiple graph convolution layers, the latent embedding of each node can contain more information from neighbours who are multi hops away. After the generation of the latent embedding of the nodes in the graph, one can either easily feed the latent embeddings to feed-forward networks to achieve node classification of regression goals or aggregate all the node embeddings to represent the whole graph. And then perform graph classification and regression. [8, 19]

1.2.3.4 RNN and LSTM.

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a sequence. RNN is designed to recognize the sequential characteristics and use patterns to predict the following likely scenario. They are widely used in speech recognition and natural language processing applications. Figure 9(a) shows the general structure of an RNN model, where X_t is the input data, A is the parameters of the network, and h_t is the learned hidden state. One can see the output (hidden state) of the previous time step t−1 is input into the neural of the next time step t. Thus, historical information can be stored and passed on to the future. [8]

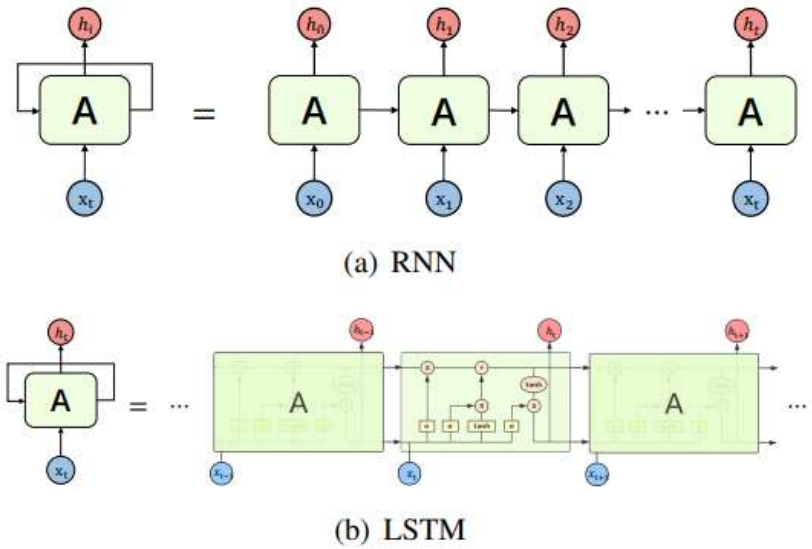


Figure 9: Structure of RNN & LSTM models [8]

A significant issue of standard RNN is that it only has short term memory due to the issue of vanishing gradients. Long Short-Term Memory (LSTM) network is an extension for recurrent neural networks, which can learn long-term dependencies of the input data. LSTM enables RNN to remember their inputs over a long period of time due to the special memory unit, as shown in the middle part of Figure 9(b). An LSTM unit comprises three gates: input, forget and output gate. These gates determine whether to let new input in (input gate), delete the information because it is not essential (forget gate), or to let it impact the output at the current time step (output gate). Both RNN and LSTM are widely used to deal with sequence and time serious data for learning the temporal dependency of the ST data. [8, 19].

1.2.3.5. Seq2Seq.

A sequence to sequence (Seq2Seq) model aims to map a fixed-length input with a fixed-length output where the input and output length may differ [138]. It is widely used for various NLP tasks such as machine translation, speech recognition and online chatbot. Although it was initially proposed to address NLP tasks, Seq2Seq is a general framework and can be used for any sequence-based problem. As shown in Figure 10, a Seq2Seq model generally consists of 3 parts: encoder, intermediate (encoder) vector and decoder. Due to the powerful ability to capture the dependencies among the sequence data, the Seq2Seq model is widely used in ST prediction tasks where the ST data present high temporal correlations such as urban crowd flow data and traffic data. [8, 19].

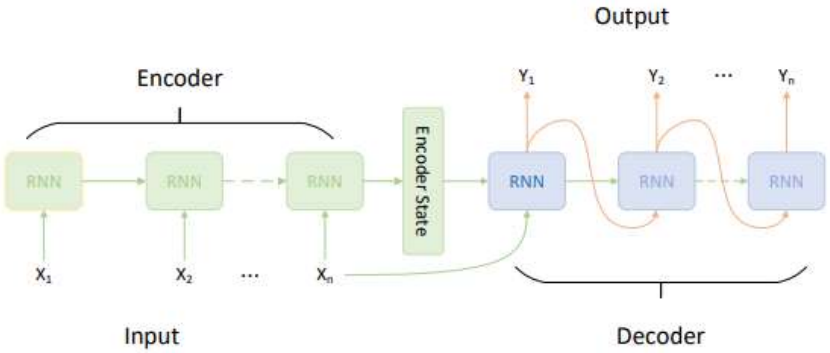


Figure 10: Structure of Seq2seq model [8]

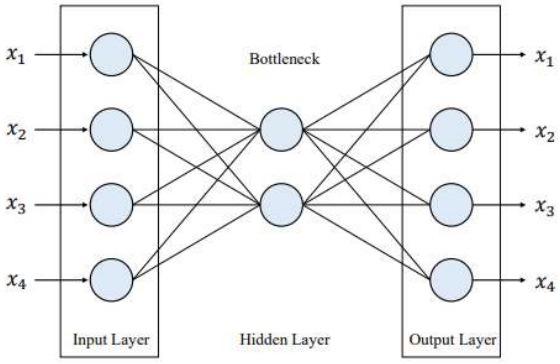


Figure 11: Structure of the one-layer AE model [8]

1.2.3.6 Autoencoders (AE)

An autoencoder is an artificial neural network that aims to learn efficient data encodings in an unsupervised manner [53]. As shown in Figure 11, it features an encoder function to create a hidden layer (or multiple layers) that contains a code to describe the input. A decoder then creates a reconstruction of the input from the hidden layer. An autoencoder creates a compressed representation of the data in the hidden layer or bottleneck layer by learning correlations in the data, which can be considered a way for dimensionality reduction. As an effective unsupervised feature representation learning technique, AE facilitates various downstream data mining and machine learning tasks such as classification, heatmap prediction, heatmap generation, and clustering. A stacked autoencoder (SAE) is a neural network consisting of multiple layers of sparse autoencoders in which the output of each layer is wired to the inputs of the successive layer [7]. In this paper, the Convolutional (CAE), the Variational (VAE), and Convo-Variational (CVAE) Autoencoders are used to train our NTPC-fire dataset of the spatial domain for heatmap generation.

1.2.3.7 Attention Mechanisms (AM)

Attention is a mechanism developed to improve the performance of the Encoder-Decoder RNN on machine translation [5]. As an unattended Encoder-Decoder encodes only limited length sequence representation, it performs poorly with long input sequences. The solution of the AM allows the model to learn which encoded-words in the source sequence to pay attention to and to what degree during the prediction of each word in the target sequence. It also works against ST-dataset too in the form of visual attention. [13, 14, 15, 18, 70, 72]. Transformers and SEQ2SEQ architectures are examples of attentive architectures.

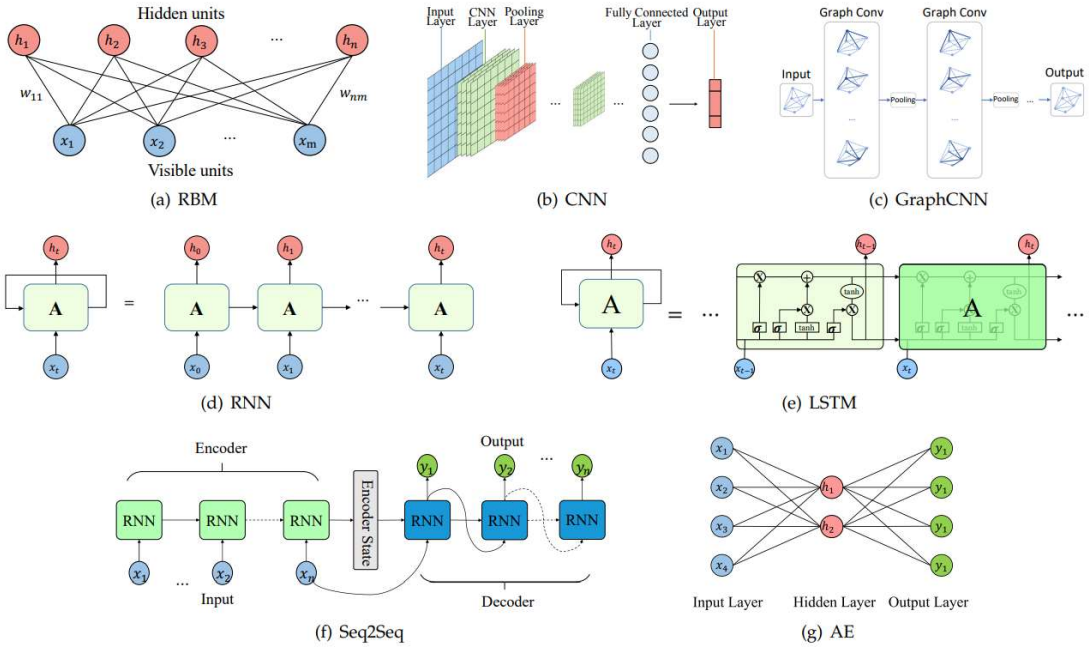


Figure 12: an overall diagram of an Artificial Neural Networks used in STDM [8]

1.2.3.8 Generative Adversarial Networks (GANs)

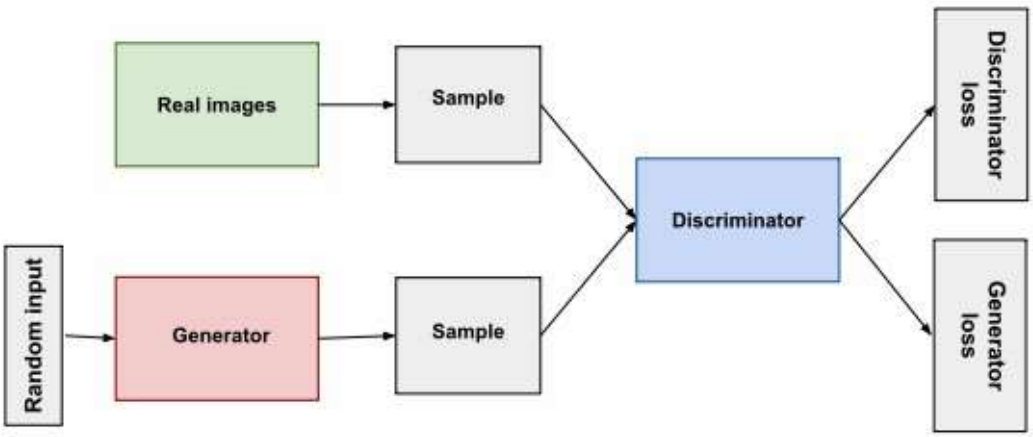


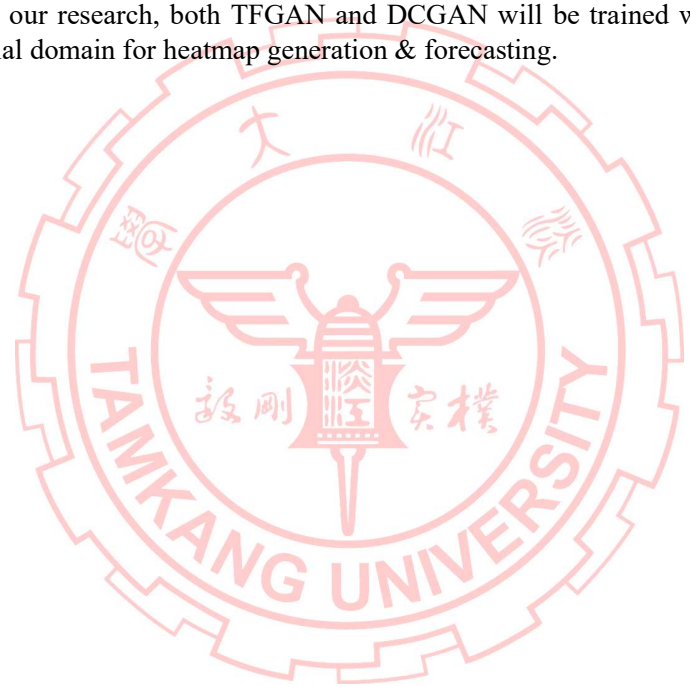
Figure 13: Overview of a GAN [75]

According to [75], generative adversarial networks (GANs) are generative models designed to create new data instances that resemble your training data. E.g., GANs can create images resembling real human faces, even though they do not belong to any real person. [75]

Structurally, GAN is divided into two sub-architectures (both are neural networks): the generator and the discriminator. The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator. Its output becomes the discriminator's input. While training, through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights to minimize both losses. [75]

The discriminator learns to distinguish the fake data (generated by the generator) from the real data (training input). The discriminator penalizes the generator for producing implausible results. While training, it produces both generator and discriminator loss: the value used to backpropagate (update weights of) their respective sub-architectures. [75]

In our research, both TFGAN and DCGAN will be trained with our NTPC fire dataset of the spatial domain for heatmap generation & forecasting.



CHAPTER 2: LITERATURE REVIEW & RELATED WORKS

2.1 Notable referred works for early STDM models

Doshi, Basu and Pang [1] developed a CNN model identifying disaster-impacted areas by comparing the change in artificial features extracted from satellite imagery. Using a pre-trained semantic segmentation model from [2] they extracted artificial features, the pre-and post-event images on the "before, during and after" imagery of the event-affected area.

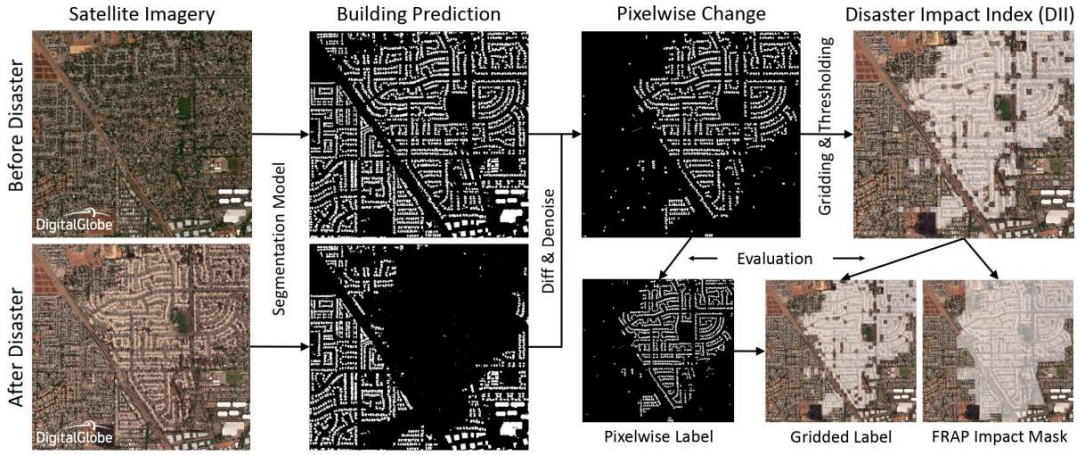


Figure 14: Doshi's Residual Inception Skipnet model for disaster insight [1]

Amit and Aoki [3] proposed a CNN consisting of a sequence of layers, the convolution layer (who detects features from a data image), the pooling layer (downsamples the input), and the FC layer (who classifies the features detected earlier), with ReLU as the primary activation function of the network. Further explained in section 2 of [3].

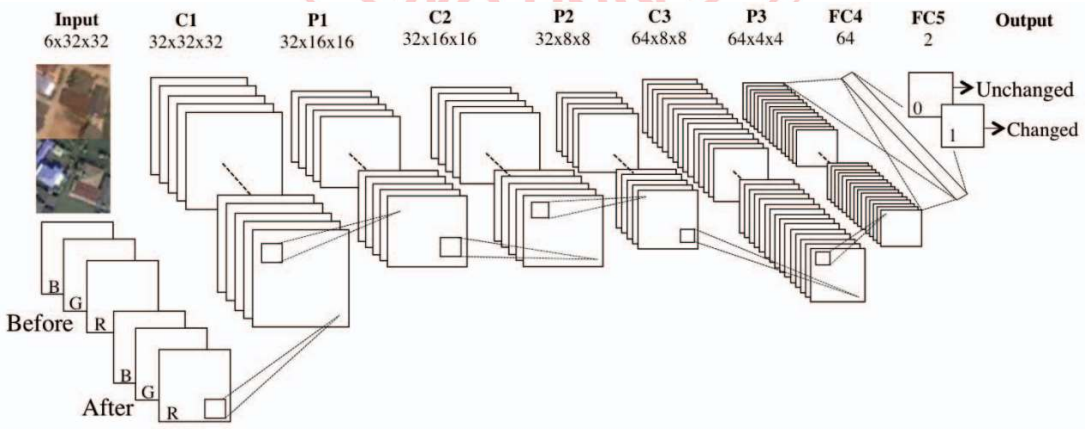


Figure 15: Amit and Aoki's CNN based disaster detection model used in [3]

Iglovikov, Mushinskiy and Osin [4] used an FC-CNN named U-NET and an embedded multispectral sensor, which detects frequency reflection by the objects to detect geo-features in satellite images and yielded satisfying results.

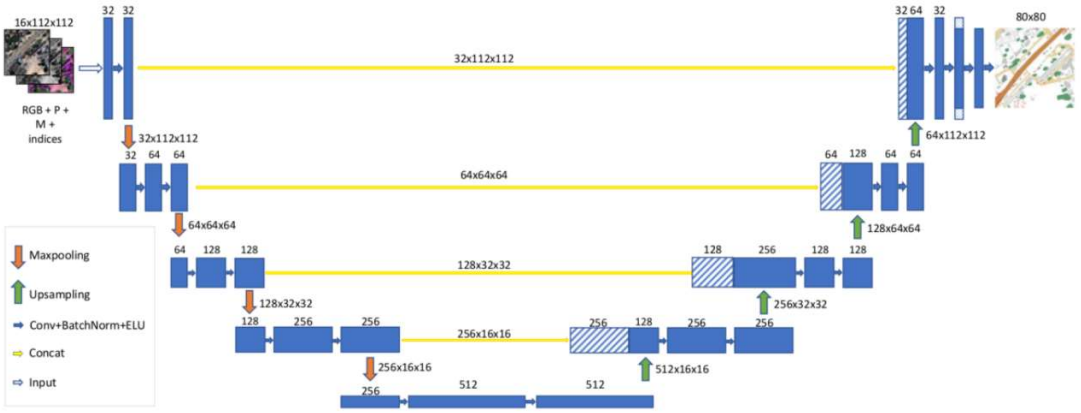


Figure 16: Iglovikov's UNet architecture for geo-feature detection, featuring the downsampling and the upsampling sections [4]

Bochkovskiy, Wang and Liao [5] incorporated YOLO V.4 and TensorFlow Keras into a CNN to improve performance in image recognition. Possibly, we can deploy such a CNN model in this thesis.

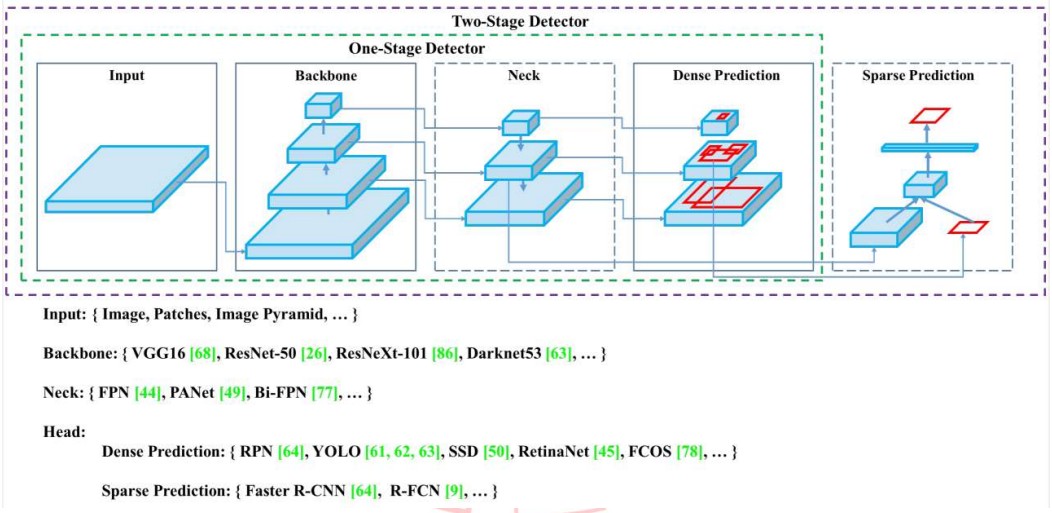


Figure 17: Bochkovskiy's Yolov4 architectural diagram used in [5]

In terms of video and sequence type photos (such as slideshow), however, the use of LSTM-RNN is needed. According to Fang et al. [6], LSTM is excellent at predicting floods because it could process time-series data. In turn, they designed the long spatial sequential LSTM (LSS-LSTM).

2.2 Notable referred works for graph-based state-of-the-art models

Li et al. [7] view spatiotemporal forecasting as a crucial task for a learning system that operates in a dynamic environment. It can be helpful in pathfinding, autonomous vehicles, logistics, city planning etc. They used a Diffusion Convolutional Recurrent Neural Network (DCRNN) model to forecast the road traffic within a specific space and timeframe (The dataset was METR-LA, 2014). Diffusion convolution extracts the traffic features, and the RNN processes the traffic volumes in sequence. [7]

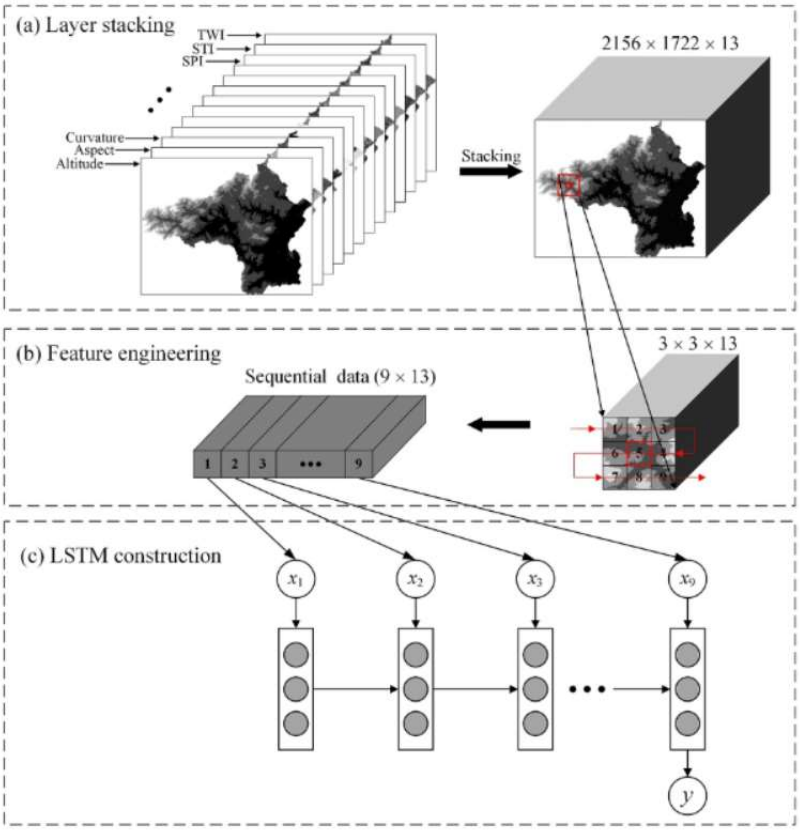


Figure 18: Mechanics of Fang's LSS-LSTM model used in [6] to predict flood in Shangyou county, Jiangxi, China

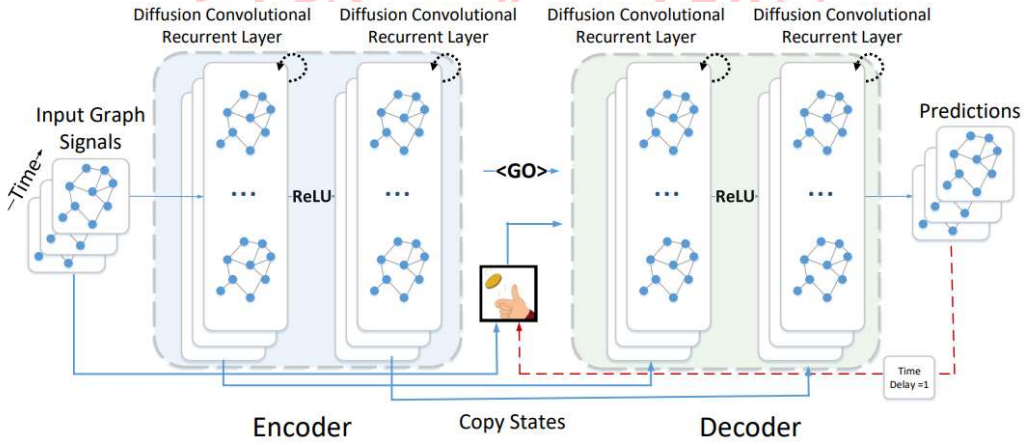


Figure 19: Mechanics of Li's DRCNN model used in [7] to predict the traffic density of each timestamp and location

Wang et al., 2019 [8] surveyed and collected several papers about spatiotemporal data mining. Moreover, they explained the fundamentals and the concepts of STDM. According to the paper and Figure 20, most of the Deep learning STDM models are used for prediction, especially traffic prediction, which comprises most of the works referred to by this paper.

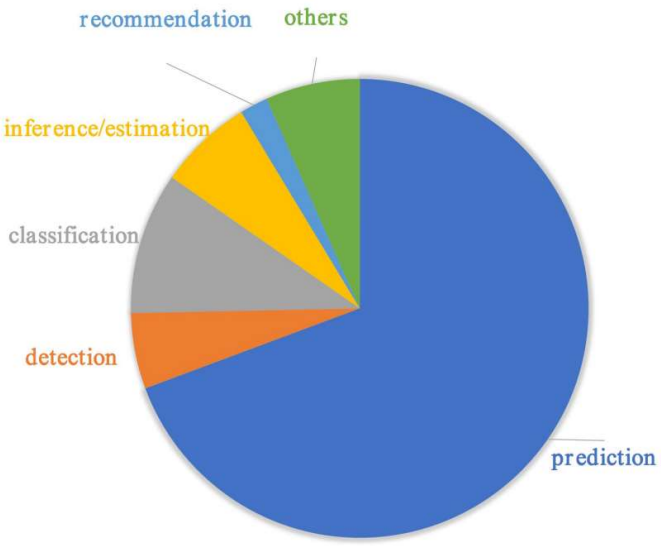


Figure 20: Distribution of the STDM problems addressed by deep learning [8]

Yu et al., 2018. [9], proposed a Spatiotemporal Graph Convolutional Network (STGCN) to tackle the time series prediction problem in the traffic domain. They formulated the problem on graphs and built the model with complete convolutional structures, enabling faster training with fewer parameters. Compared with existing models, STGCN more effectively captured comprehensive spatiotemporal correlations through modelling multi-scale traffic networks and consistently outperformed state-of-the-art baselines on various real-world traffic datasets. [9]

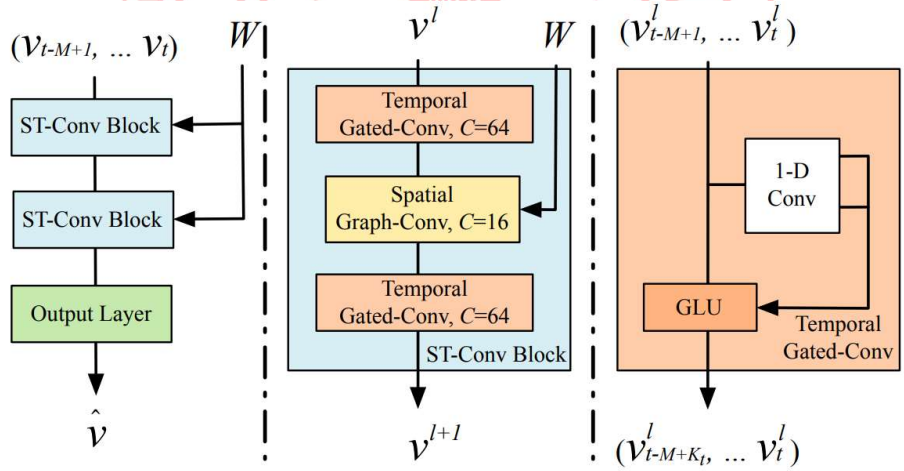
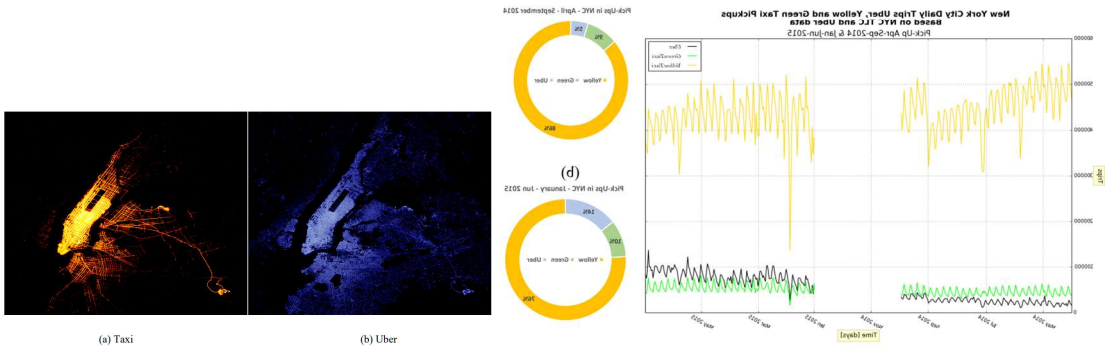


Figure 21: Yu's architecture of spatio-temporal graph convolutional networks – the full mechanism is described in the 3rd section of [9]

Correa et al., 2017 [10], performed a spatiotemporal data mining of Taxi vs Uber ridership in NYC, 2014+15. According to Figure 22(a), the ridership for both taxi systems depended on several factors – such as personal income, education, jobs, car ownership etc. With three spatial models for ridership prediction – linear, spatial error and spatial lag models, the last one outperformed the first two algorithms and yielded considerable accuracy and performance [10].



(a) (b)

Figure 22: Correa's Taxi-Uber dataset visualization. The spatial module is displayed as heatmap (19a) which is used to predict hailing densities. Meanwhile, the temporal module is displayed as line graph (19b), which is used to predict future ridership volume

Amato et al. [11] designed a deep learning-based architecture called "Empirical Orthogonal Functions principal component analysis" EOF-PCA in which the EOF framework decomposes the spatiotemporal input data, in terms of a sum of products of temporally referenced basis functions and of stochastic spatial coefficients which can be spatially modelled and mapped on a regular grid. Then, the input layer spatial covariates are processed by a "Fully Connected Neural Network" (FCNN) to obtain predictive coefficients to be recomposed altogether with the decomposed data stream, obtaining a spatiotemporal signal reconstruction. [11]

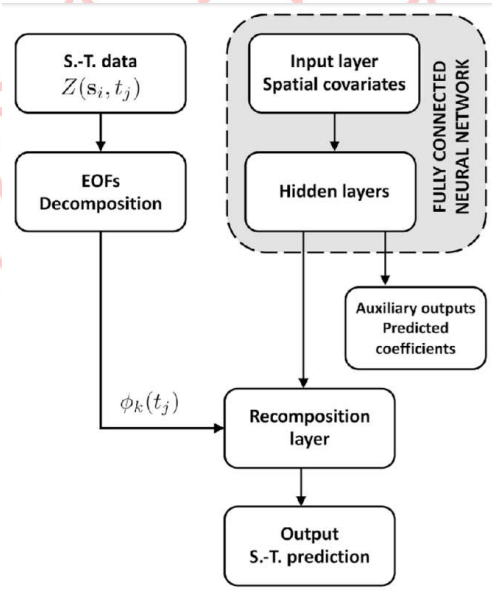


Figure 23: Amato's architecture. The temporal bases are extracted from a decomposition of the S.T. signal using EOFs. Then, an FCNN is used to learn the corresponding spatial coefficients [11]

The short-term forecast of rail transit is an issue in intelligent transportation systems (ITS). An accurate forecast can forewarn travel outbursts, helping the passengers with their travel plans. Tang et al. [12] designed an LSTM based framework to learn and forecast rail traffic. Even though the LSTM is notably effective in temporal data, it cannot correlate the time domain with the space domain, causing them to propose the ST-LSTM architecture. Compared with other conventional models, the ST-LSTM network can perform better in experiments. [12].

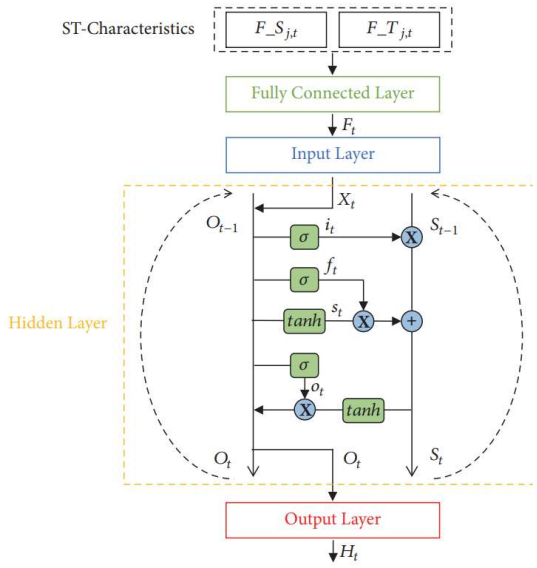


Figure 24: Tang's ST-LSTM architecture [12]

Lu et al. [13] designed a spatial-temporal deep learning network, termed ST-TrafficNet, for traffic flow forecasting, and whose architecture works as follows. 1. The Spatial Aware Multi-Diffusion Convolution Bloc (ADC-Block – who introduces Graph Attention Mechanism (GAM) into the MDC) uncovers unseen spatial dependencies from traffic graph signals automatically 2. The multi-diffusion convolution (MDC) block harvests ST-features of the spatial domain from the data stream. 3. The ST-TrafficNet, an LSTM based framework, harvests the features of the temporal domain. 4. The output from both ANN are summed up to achieve convolutional results. And 5. The ST-TrafficNet is evaluated on two benchmark datasets and compared with various baseline methods for traffic forecasting. [13]

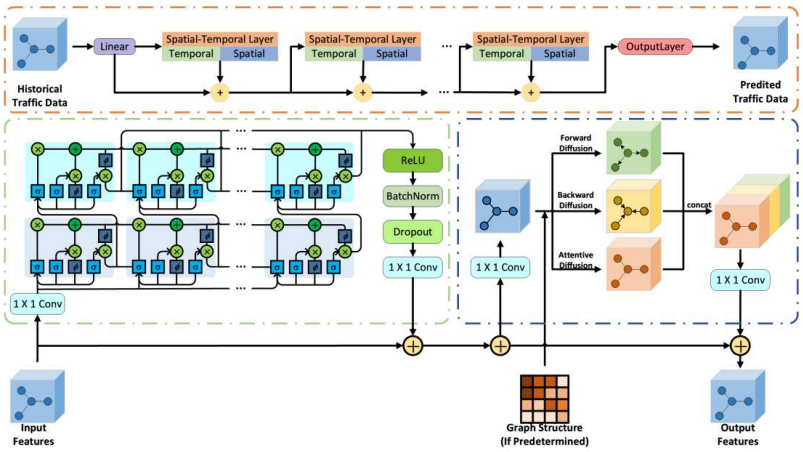


Figure 25: Lu's ST-TrafficNet architecture for traffic flow prediction [13]

Pan et al. [14] designed a deep learning framework for traffic flow prediction called "ST-Metanet". According to them, traffic prediction enhances traffic safety and enhances the ITS. However, it must face two challenges: 1) complex spatiotemporal correlations of urban traffic and 2) diversity of such spatiotemporal correlations. To tackle these challenges, they designed the ST-MetaNet model to predict urban traffic in all locations collectively. ST-MetaNet employs a seq2seq architecture, consisting of an encoder to learn historical traffic information and a decoder to make

predictions step by step. More specifically, the encoder and decoder have the same network structure, which contains a recurrent neural network (RNN) to encode the urban traffic, a meta graph attention network (Meta-GAT) to capture diverse spatial correlations, and a meta recurrent neural network (Meta-RNN) to consider diverse temporal correlations. Extensive experiments were conducted based on two real-world datasets to illustrate the effectiveness of ST-MetaNet against several state-of-the-art methods.[14] Since the ST-Metanet model is our validating basis, we will put referred results directly in our comparison experiment [14].

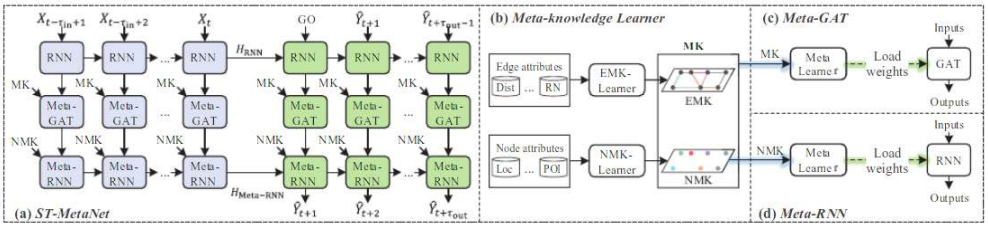


Figure 26: Pan's ST-MetaNet architecture for traffic flow & speed prediction [12]

De Medrano et al. [15] designed A Spatio-Temporal Spot-Forecasting Framework for Urban Traffic Prediction, named CRANN (Convo-Recurrent Attentional Neural Network). It is highly adaptable in several ST conditions, easy to understand and interpret, and better & more stable than state-of-the-art alternatives. [15]

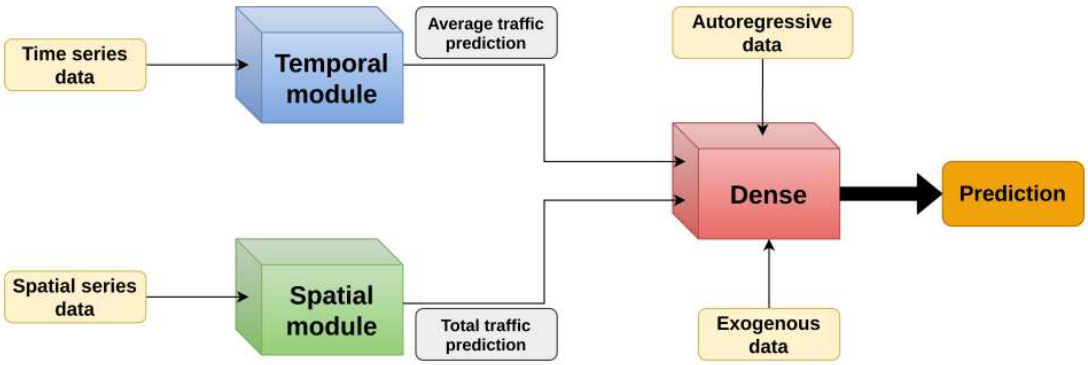


Figure 27: De Medrano's schematics for the CRANN architecture (a) [15]

To cope with the nonlinearity of the traffic flow data during the holidays, Luo et al., 2019 [16] designed a discrete Fourier transform (DFT) and support vector regression (SVR) based machine learning model to predict the road traffic flow during the holidays in Jiangsu Province, China, on Tomb-sweeping Day and National Day from 2011 to 2015. Properly trained, it outperformed other ML models like ARIMA, SVR, and EMD-SVR. The model is described in the paper itself [16].

Shih et al. 2018 [18] designed an LSTM capable of processing multiple time series at the same time called "Temporal Pattern Attention LTSM" (TPA-LSTM). The architecture is designed to process complex and non-linear interdependencies between time steps of multivariate time series data. An RNN with an attention mechanism is designed and deployed to learn long-term dependency in time series data for an accurate prediction. The typical attention mechanism reviews the information at each previous timestep and selects relevant information to help generate the outputs; however, it fails to capture temporal patterns across multiple timesteps. The model uses a

set of filters to extract time-invariant temporal patterns, like transforming time-series data into its "frequency domain". The attention mechanism selects relevant time series and uses its frequency domain information for multivariate forecasting. Surprisingly, regardless of the cases, the model achieved comparable performance with other state-of-the-art models and architectures. [18]

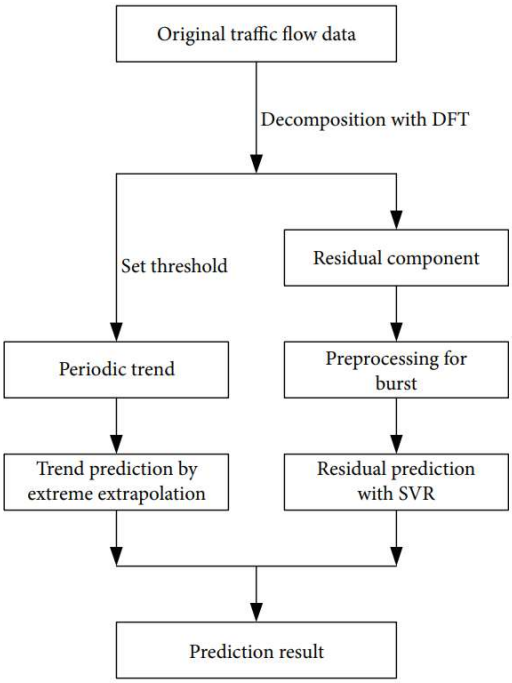


Figure 28: De Medrano's schematics for the CRANN architecture (b) [15]

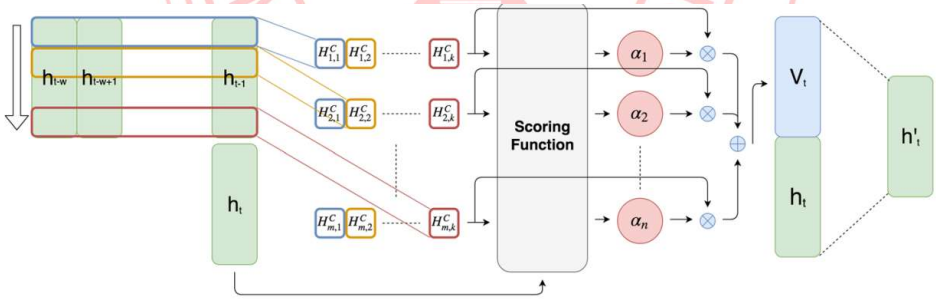


Figure 29: Shih's Attentive RNN architecture for multivariate-time series prediction [18]

Atluri, Karpatne, and Kumar, 2017 [19], like Wang et al. in [8], restated and emphasized the concepts of spatiotemporal data mining.

Bai et al. (2020) [71] designed a traffic forecasting model named "Adaptive Graph Convolutional Recurrent Network" (AGCRN), whose data is gathered from Los Angeles Traffic Data (PEMS04 & 08). With GCN sub-architecture implemented, the model is capable of handling spatial and temporal correlations of the traffic flow, speed, and volume. The AGCRN could automatically capture fine-grained spatial and temporal correlations in traffic series based on the two modules and recurrent networks. Furthermore, the model outperformed several state-of-the-art models significantly without pre-defined graphs about spatial connections. [71]

Guo et al. (2019) designed a traffic forecasting GCN based architecture named "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting" (ASTGCN) [72] and the earlier Attention-based Spatial-Temporal Graph Neural Network (ASTGNN) [83]. With non-GCN models incapable of modelling dynamic ST-correlations of the traffic data, the prediction is somewhat inaccurate without graph analysis. Both models comprise of 3-Independent components; recent, daily, and weekly dependencies; and 2-submodels: 1—ST-Attention Mechanism to capture ST-correlations in the data 2. ST-Convolution simultaneously employs graph convolutions to capture spatial patterns with the typical convolutions describing temporal features. Then, the weights generated by these dependencies are fused to generate prediction results – outperforming most existing state-of-the-art models. The dataset was Caltrans Performance Measurement System (PeMS) [72].

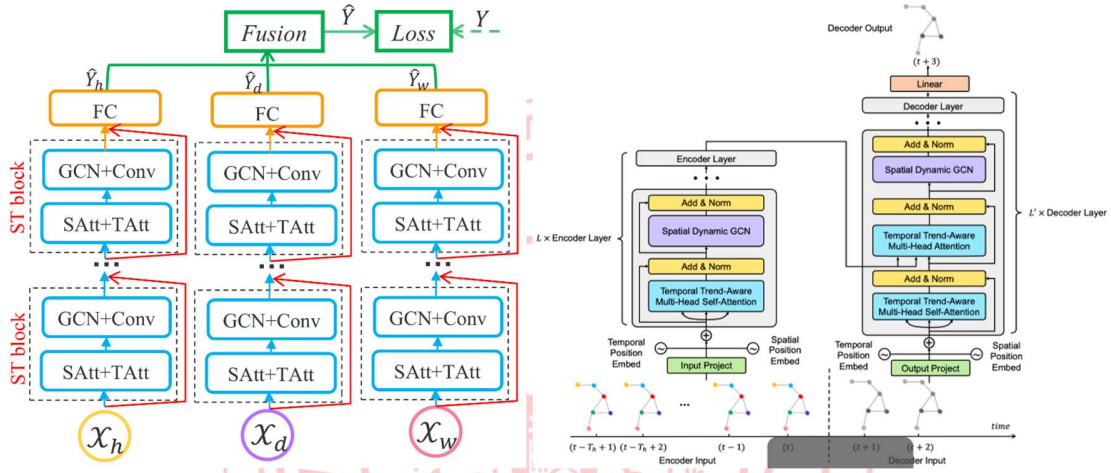


Figure 30: Guo's ASTGCN [72], left and ASTGNN [83], right

Ghaderi et al. (2017) [73] designed a serial multi-LSTM framework named "Deepforecast – DL-Spatiotemporal Forecasting (DL-STF)". In that paper, they modelled the spatiotemporal information by a graph whose nodes are data-generating entities, and its edges basically model how these nodes interact. The model can simultaneously forecast all nodes in the graph based on one framework. The dataset is MS_winds.dat, the collection of windmills in the northeast USA. Compared to other benchmark models, the DL-STF was excellent at short term forecasts. [73]

For attentive models, Grigsby, Wang, and Qi (2021) [80] designed a transformer-based STDM architecture by connecting an attentive seq-seq model (for the temporal domain) to an attentive GCNN (for the spatial domain) for multivariate ST-data forecasting on the NY-TX weather and the METR-LA traffic datasets and achieved outstanding results. Unfortunately, in our comparative experiment, the model did not make it to the testing phase after being trained for 80 epochs (8 hours in a 4x GPU TWCC container) due to a GPU memory burst, causing it to be another basis. [80]

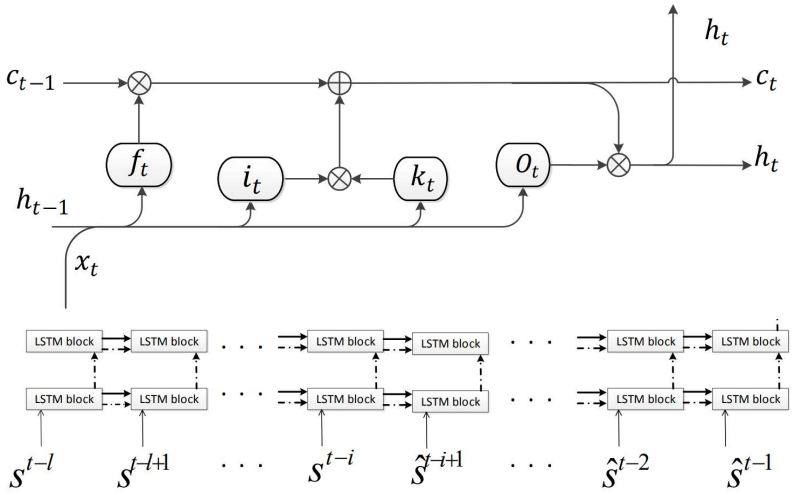


Figure 2. The model M_i trained at time t. Inputs are $s^{t-\ell}, s^{t-\ell+1}, \ldots, s^{t-i}$ (real values) and $\hat{s}^{t-i+1}, \ldots \hat{s}^{t-1}$ (forecasted values from previous trained models). Black thick and dashed arrows are c_t and h_t based on Figure 1.

Figure 31: Ghaderi's DeepForecast Multi-LSTM model [73]

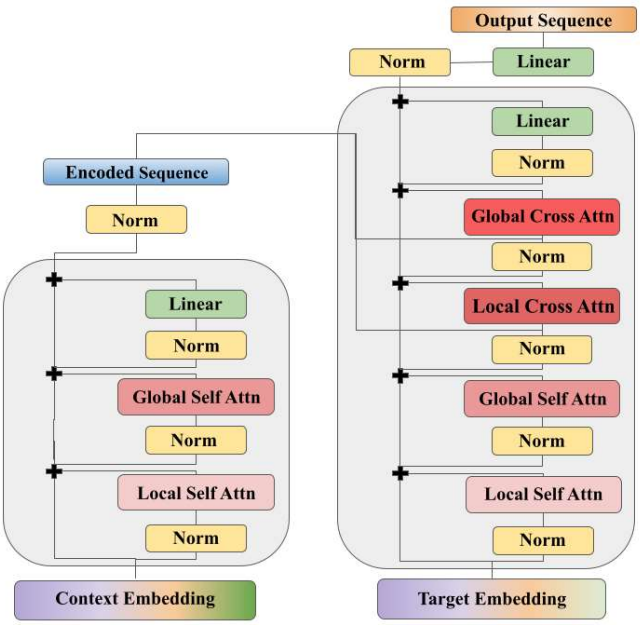


Figure 32: Grigsby's Spatiotemporal Transformer model [80]

2.3 Notable referred works of models suiting non-graph event data

To apply the model design process to non-graphical event data, such as dot-and-hotspot event datasets, we must first split the dataset into the spatial domain (rasterised heatmaps) and the temporal domain (time series of frequency by timeframe). Most of the models are written in Python with the Tensorflow-Keras library.

For the spatial domain, the most fundamental models used in our paper as a reference for the study are from the TensorFlow-Keras website [74] and Brownlee's blog [84]. Including Autoencoders [74] and GANs [75]. Most of our generative architectures are from this source: [78], notably DCGAN [79], LSGAN [82] and Wasserstein GAN (WGAN) [74]. Most of the spatial models referred are designed based on the MNIST (28*28) dataset, prompting to rasterise the spatial domain to an album of 28*28 images.

For the temporal domain, the LSTM-RNN model is the most popular. Moreover, Facebook Prophet (FBProphet) [76] is also a powerful model for small datasets, rivalling the LSTM in terms of performance metrics. For Example, Menculini et al. [78] designed a wholesale food price prediction model by comparing ARIMA, FBProphet, LSTM, and CNN-LSTM in their performance. Due to the data's apparent seasonality, they found out that FBProphet did not perform well as expected, while ARIMA rivalled a simple LSTM – with CNN-LSTM model outperforming them all due to feature pre-extraction functionality of the CNN. In the future, if we have a chance, we will re-implement and carry out experiments on Attentive models, namely ST-transformers of [80].

2.4: Applications of Deep Learning Models in STDM

Restating what we have said in Section 1, the paper [8] and [19] covered several fields of applications, such as climate science, neuroscience, social sciences, epidemiology, transportation, criminology, and Earth sciences. Here are some GIS application examples:

- **Meteorology:** Prediction of poor weather and disasters assist in reducing life and property loss. For example, the use of CNN is found in [1], [2], and [3]; as well as LSTM found in [6], as well as Zhang et al.'s, 2018 [48], designed a Multilayer Perceptron Neural Network (MLPNN) for short term rainfall forecasting in China and achieved considerable performance – outperforming ARIMA and SVM.
- **Transportation:** Corresponding to Figure 20, STDM has the most popular application in transportation. With the rising availability of transportation data collected from various sensors, there is an urgent need to utilize deep learning methods to learn the complex and highly non-linear spatiotemporal correlations among the traffic data to facilitate various tasks. Traffic flow, direction & speed prediction is the most popular application of this field, e.g. [7], [9], [12], [13], [14], [15], and [16].
- **On-Demand Service:** Taxi-hailing apps are becoming popular. One of the notable works for taxi usage prediction is [10].
- **Human Mobility:** Mining the human mobility data is practically important for applications, including traffic forecasting, urban planning, and human behaviour analysis. Many recent works use deep learning methods for urban crowd flow prediction and crowd density estimation; for example, Zhang et al. 2018 [49] designed an ST-ResNET based model to predict citywide crowd flow in Guiyang, Beijing, and New York City.
- **Location-Based Social Networks (LBSNs):** Some social network platforms, such as Foursquare and Flickr, use GPS features to locate the users and let the users broadcast their locations and other contents from their mobile device. Various deep learning models are currently designed to analyze the user-generated ST data in LBSN; for example, Zhao et al. 2018 [50] designed an ST-LSTM model to recommend POIs for Foursquare and Facebook users.
- **Criminology:** A criminal ST data consists of location coordinates and timestamps. DL models are set to predict crime incidence on a heatmap; for example, Duan et al. [51] proposed a Spatiotemporal Crime Network based on CNN to forecast the crime risk of each region in the urban area for the next day.

2.5: Summary of section 2

Furthermore, we also categorized referred papers into several application categories and discovered that most of them belong to the traffic prediction category. To sum up, most models referred to, designed for traffic prediction work, are built with complex DL architectures, which improved accuracy and reliability over basic models. DL-based architectures also contribute to the data mining of different learning approaches, which are mentioned in section 3.1. Nevertheless, to apply the DL models on our non-graph dataset, we need to implement models referred to in section 2.2 in our second experiment, the custom event heatmap experiment (CEHE).

CHAPTER 3: METHODOLOGY

In this section, first, we will list different approaches to STDM. Then, we will discuss the advantages and disadvantages of those approaches. Then, to apply the STDM process to our NTPC-fire dataset (a dot-and-hotspot event dataset), and choose some suitable models for prediction, and for future research, classification. After the discussion, different performance measures for STDM will be compared and we will try to address the effects of those measures. We will conclude this section with characteristics for different STDM processes. Everything related to this paper, especially experiment resources, has already been saved in our GitHub repository[1].

3.1 Different approaches to spatial-temporal data mining

As the extension to Section 2, in some studies, such as [8] and [19], ST-Datasets were mined using both traditional Machine learning methods, such as Decision tree, DBSCAN, SVM, ARIMA, etc., and deep learning methods – CNN, RNN, LSTM, Transformers, etc. As listed below, we summarize different approaches to ST-datasets.

- **Clustering:** Clustering refers to grouping instances in a dataset that shares similar feature values. **In STDM, clustering can be performed on:** Points, trajectories, time-series, spatial maps, and ST raster data**. Nevertheless, a challenge must be overcome;** clustering locations based on their time series must ensure that the discovered clusters are spatially contiguous. Ignoring this can lead to location data misinterpretation. There are **some notable clustering algorithms**, **namely** ST-DBSCAN [20], clustering ST-points using DBSCAN algorithm [21], CLARANS [22], 'dynamic ST clusters' [23]. Etc.
- **Prediction:** Predictive learning learn a mapping from the input features to the output variables using a representative training set. In spatiotemporal applications, both the input and output variables can belong to different types of ST data instances.
  - **Time-series:** RNN and LSTM are widely used for time-series data prediction. The weather variables such as wind speed are usually modelled as time-series, and then RNN/LSTM models are applied for future weather forecasting [24], [25], [26], [27], [28], [29].
  - **Spatial maps:** The spatial maps can be usually represented as image-like matrices and thus are suitable to be handled by CNN for various predictive learning tasks [30], [31], [32], [33]. One of the examples involving spatial map prediction is of Zhang et al., 2016 [32]. They proposed a CNN based crowd flow forecasting model called UrbanFlow for real-time urban crow flow prediction. In order to capture the temporal and spatial correlations of a sequence of spatial maps simultaneously, many works tried to combine CNN with RNN for the prediction. This method is utilized in [6], [7], [14], [15], and [18]. Generative models dedicated to rasterized heatmaps [74], such as Autoencoders and Generative Adversarial Networks (GANs), are preferred in our work to predict a heatmap of a disaster, such as building fires.
  - **Trajectories:** Currently, two types of deep learning models, RNN and CNN, are used for trajectory prediction depending on trajectorial data representations. Trajectories are a sequence (learnable by RNNs) type data of locations representable as a matrix (learnable by CNNs).]. One of the examples involving trajectorial prediction is of Lv et al., 2018 [34]. They modelled trajectories as two-dimensional images, where each pixel represented trajectorial locations. Then an MLCNN were adopted to combine multi-scale trajectory patterns for destination prediction of taxi trajectories. Furthermore, some works also incorporate GPS, such as [35], who proposed a 4-layer LSTM model named DeepTransport, learning a set of GPS trajectories, to predict different transportation modes
  - **ST Raster:** ST raster can be represented either as matrices dimensioned as location and time or 3D tensors dimensioned as cell region ID, cell region ID, and time. [8] Usually, 2D-CNN (matrices) and 3D-CNN (tensors) are applied for ST Raster data prediction, and sometimes they are also combined with RNN. 3D-CNN models are purposed in [36], [37], [28] and [38]. For

[1] Available in github.com/Suppersine/Thesis2021/

example, Rasp & Lerch, 2018 [28], modelled the mobility events of passengers in a city in different time slots as a 3D tensor, then they used a 3D-CNN model to predict the supply and demand of the passengers for transportation. The major difference between an ST Raster and a spatial map is that the ST raster is the merged measurements in multiple time slots, while the spatial map is the measurement in only one timeslot. While preparing our NTPC-fire dataset, we rasterised our spatial map with dot data by loading event dots into each raster cell to obtain their concentration using our program's "cytogenesis" function.

- **Using Temporal Information (Classic ML) [19]:** There are some estimations of time-series' future trends, such as exponential smoothing techniques (ESTs) [39], ARIMA models [40], and state-space models [41].
- **Using Spatial Information (Classic ML) [19]:** We surveyed a vast body of literature on spatial prediction methods that consider the spatial auto-correlation structure in the data to ensure spatially coherent results, such as spatial autoregressive (SAR) models [42], geographically weighted regression (GWR) models [43], and Kriging [44] Markov random field-based approaches [45], [46], [47].

- ***Representation Learning:*** *RL-models* aims to learn the abstract and useful representations, formed by (non)linear transformations compositions of input data, of the input data to facilitate downstream data mining or machine learning tasks
  - **Trajectories**: Trajectories are ubiquitous in location-based social networks (LBSNs). CNN, GCN, GRU and RNN are used to learn such representations. For example, Ding et al, 2018 [52], proposed a geographical convolutional neural tensor network named GeoCNTN to learn the embeddings of the locations in LBSNs.
  - **Spatial maps:** There are several works studying how to learn representations of spatial maps. CNN, GCN, and Autoencoders are used in this learning method. For example, Costilla-Reyes et al., 2017 [53] proposed a CNN architecture for learning ST features from raw spatial maps of the sensor data.
- **Classification:** The classification task is mostly studied in analyzing fMRI data for disease identification. In Neuroscience, for instance, [55] employed an LSTM to diagnose autism spectrum disorders (ASD) using fMRI time-series data. Furthermore, [56], [57], [58], [59], [60], [61] modelled the fMRI data as spatial maps and then used them as the input of the classification models. Feature classification & recognition is also present in [3], [4], and [5], which used a CNN based architecture to classify features in satellite images.
- **Estimation and Inference:** Current ST-data estimation and inference works mainly focus on spatial map and trajectory data types.
  - **For spatial maps,** while monitoring stations have been established to collect pollutant statistics, the number of stations is usually minimal due to the high cost. One of the models utilizing this method is the ADAIN architecture, which [62] applied for modelling air quality inference of any location-based on pollutants and learning some complex feature interactions.
  - **For trajectories,** STDM of this purpose is used along with GPS, which is notable in [10]. Another example is of Zhang et al., 2018 [63]. They proposed an RNN-based deep model named DEEPTRAVEL, which can learn from the historical trajectories to estimate the travel time.
- **Anomaly Detection:** Anomaly detection or outlier detection aims to identify the rare items, events or observations that differ remarkably from the rest of the data. For instance, ST-DBSCAN is used in ST point anomalies [64], [65] and [66]. Moreover, Anomaly detection is also used in disaster analysis & prediction in [1] and [6]. DBN and LSTM are used for event data on traffic accident tweets [67]. For example, Zhu et al., 2018 [68] proposed utilising Convolutional Neural Networks to automatically detect traffic incidents in urban transportation networks by using traffic flow data. By the way, **for spatial maps,** ST-CNN is implemented in [69] for extreme weather events identification.
- **Other notable methods:** there are some other applications, such as change detection [19], pattern mining [8], relation mining [8], [19], and POI recommender systems, [8] whose example is that Wang et al., 2019 [70] proposed an attention-based-RNN for personalized route recommendation.

3.2 Advantages and disadvantages of applying different approaches

Here, we will explain why deep learning models are preferred in recent works over traditional machine learning models, which can be seen more in Table 1:

Table 1: Comparison of DL vs ML methods in STDM. Source: [8]

	Deep Learning Models	Traditional ML models
ST Features	Automatic Learning	Hand-crafted
ST Data types	n-to-n	1-to-1
STDM Tasks	Prediction, classification, estimation, etc.	Prediction, classification, estimation, clustering, frequent pattern mining, change detection, etc.
Temporal dependency	Long/Short term	Short term
Spatial dependency	Global/Local	Local
Interpretability	Low	High
Domain Knowledge	Little	Much

According to Table 1, we conclude that DL models are chosen over ML ones due to...

- **Automatic feature representation learning:** Unlike ML models, which can only handle hand-crafted ST-features, DL models can automatically learn hierarchical feature representations from the raw ST-data. Multilayer convolution in CNN and recurrent structure in RNN can effectively learn highly complex ST-data.
- **Powerful function approximation ability:** Each DLNN has multiple layers; consisting of nonlinear modules with convolution, pooling, dropouts, and activation functions. With the composition of enough such transformations, very complex functions can be approximated to perform difficult STDM tasks with complex ST data.
- **Performing better with big data:** Traditional ML methods, e.g., SVM, Decision Tree, perform better on small datasets but become ineffective when the training dataset enlarges. However, DL models improve their performances when they learn larger datasets due to their powerful feature learning and function approximation abilities.

3.3 Different measures of performances

Usually, each DL framework has its performance measured in mean squared error (MSE), mean absolute error MAE), and root mean squared error (RMSE). Some models, however, used different performance metrics, such as that of Shih et al. 2018 [18], which had some unfamiliar metrics, like RAE, RSE, CORR, etc. Thus, his code needs an overhaul to ensure that both the input and the output tensor dimensions are equal; to calculate MSE. The experiment, divided into two episodes, has been further described in section 4.

In our first episode of the experiment, from each referred/surveyed paper, primarily on Google-COLAB environment with TWCC as the backup environment, which, if a model exceeds the COLAB's resource limits, we will run its Python code there. Then, using their original datasets, we will run the models we have referred to in section 2.2 and compare them in the form of MSE, RMSE, and MAE metrics; the less of them for each model, the better it performs in the term of learning and forecasting. Finally, we will conclude which model does the best learning/forecast. As this is a survey thesis, we will not design a custom model.

In our second episode, given an event map NTPC-fire dataset, we will visualize and split the dataset into the spatial (by converting it into a raster heatmap) and the temporal domains (by converting it into a time series), and choose some familiar yet straightforward models to learn and forecast the data, and do the analytics same as the first episode.

3.4. Hypothesis:

To hypothesise our research, we have asked ourselves two primary questions:

3.4.1. STDM DL-models can do a variety of learning and predictions, but how well can they do?

- Metrics: MSE, RMSE, & MAE
- What architecture is the best model in STDM?
- Is the ST-METANET model still the best?
- Do newer, more architecturally advanced models always outperform the older ones?

3.4.2. If we have a custom dataset with its data structure visualised, which model to be learned from it?

- Such as the "NTPC-fire_2015-17" dataset which is a small event dataset
- Do we need to design a new model?
- If we implement existing models to train such a dataset, which model makes the best prediction for each domain?
- Of the same model, what will happen if we vary some hyperparameters?

3.5. Contributions

To be seen in section 4, we have studied several STDM papers and proved their model interesting and applicable. Furthermore, we have run five models: Keras-LSTM, CRANN, DCRNN, Multi-LSTM and ST-METANET; they all performed well for their architectures. More contributions can be read in Section 4.4.

Also, we ran generative models, like AEs and GANs, to predict our rasterized map and predicted the trends using time-series forecasting models (like LSTM-RNN, FB-prophet, and ARIMA), along with performing some subtests which will eventually lead to hyperparameter tuning.

CHAPTER 4: EXPERIMENTS

4.1. Experiment Settings for Referred Papers (a.k.a. Exploratory Comparative Experiment; ECE)

4.1.1 ECE Settings

Our experiment[2] is divided into two episodes; one is for referred models for comparison called "exploratory comparative experiment", abbreviated ECE, and another one is of our "New-Taipei-City-fire" dataset for heatmap generation and incident prediction called "custom event heatmap experiment", abbreviated CEHE. The list below displays potentially useful STDM models used in the first episode.

Table 2: Models and datasets in the comparative experiment – brief table

Architecture	**Dataset**
LSTM-Pytorch	Taxi-Uber Dataset (NYC, 2014-15)
LSTM-TF/Keras	Taxi-Uber Dataset (NYC, 2014-15)
CRANN	Madrid traffic & weather dataset (2018-19)
ST-METANET (Flow)	Taxi-Flow (Beijing, 2010)
ST-METANET (Speed)	METR-LA (Los Angeles, 2014)
DCRNN	METR-LA & PEMS-BAY
AGCRN	Caltrans PEMS04&08
ASTGCN	Caltrans PEMS04&08
STGCN	METR-LA & PEMS-BAY
Spacetimeformer	METR-LA

Continuing from section 3.3, the deep learning models and ANNs are written in Python with Pytorch, Mxnet, and Tensorflow-Keras as our preferred DL libraries. Be advised; not every model is runnable on the stock COLAB environment and thus must be modified to meet their requirements. If a model consumes more computing resources than what COLAB can provide, it is advised to connect COLAB with a cloud VM instance or a physical Linux machine with RAM around 64 GB, for example, an EC2 Linux instance with GPU. In our case, the viable backup environment is TWCC. All lost functions are tied to their architecture, with formulae explicitly written in their referred papers.

For the Taxi-Uber-Simple LSTM models, which process only the temporal domain, the Keras simple LSTM is in the I-H-O formation of 1-4-1 (same as one in our custom dataset experiment), whilst the Pytorch simple LSTM has the formation of 1-2-2.

Only in this episode, for each model, the dataset used to train it is the same as their respective original papers. Their hyperparameters would be the same as their default settings unless changed, with Adam as their optimiser. The primary metrics used to measure the models' performance are MAE, MSE and RMSE. The models' settings are defined in Table 3. The blue characters denote models lacking a suitable environment, whose results must be directly referred from their papers.

[2] Available in github.com/Suppersine/Thesis2021/tree/main/notebooks

Table 3: Models and datasets in the comparative experiment – plus settings

Architecture	Architecture Description	Dataset Name/Desc	learning rate	Epochs	Major DL Module
Taxi-Simple-LSTM-Pytorch	Simple-LSTM	Time-series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Uber-Simple-LSTM-Pytorch	Simple-LSTM	Time-series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Taxi-Simple-LSTM-Keras	Simple-LSTM	Time-series of Taxi-Uber DS. (2014-16)	0.001	100	Tensorflow Keras
Uber-Simple-LSTM-Keras	Simple-LSTM	Time-series of Taxi-Uber DS. (2014-17)	0.001	100	Tensorflow Keras
CRANN-Temporal	Bahdanau Att.Mech Autoencoder (LSTM based)	temporal time-series of hourly/daily car traffic (in Madrid)	0.01	200	Pytorch
CRANN-Spatial	CNN+ST-Att.Mech	graph data captured by 30 sensors + Timestamps (A 17000x30 matrix)	0.01	200	Pytorch
CRANN-Dense	Fully Connected Feedforward NN (FCFFNN)	dense 3D+ tensor of both preceding modules	0.01	200	Pytorch
Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
GAT Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
ST-Metanet (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
GAT Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
ST-Metanet (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
AGCRN	Attentive Graph CRN	Caltrans PEMS04&08	0.003	100	Pytorch
ASTGCN	Attention Based GCN	Caltrans PEMS04&08	0.001	80	Pytorch
Deepforecast	Multi-LSTM	MS_winds - Wind Speed & Flow Dataset	0.001	80	Tensorflow Keras
DCRNN	R-CNN	PEMS & METR-LA	0.001	100	Tensorflow Keras
STGCN	Graph-CNN	PEMS & METR-LA	0.001	50	Pytorch
Spacetimeformer	Transformer opted for ST-data	METR-LA	0.001	80	Pytorch

4.1.2. ECE Results.

In Table 4, the green cell indicates the best model to date. The yellow highlights indicate partially successful models. The blue characters indicate possible metric values as if whose models were operational (a.k.a. basis), referenced by the numbers, which can be turned black later if a suitable environment is found. The black characters indicate models run with COLAB. The red characters indicate models run in TWCC containers. The pink cells indicate models with memory problems – insufficient RAM of either the mainboard or GPUs – requiring a suitable (64+GB RAM) environment to run. Despite stringent environmental restrictions preventing such colour-marked models from running on COLAB, it was evident that the models performed well, at least for their architectures. To see the detailed issues and viable solutions, see section 4.4.

The AGCRN, the STGCN, and the ASTGCN models suffered a serious dataset error, even if running on COLAB Nevertheless, we managed to run the model successfully with intensive “debugging and troubleshooting” noted in our "ipynb" notebooks. With a significant overhaul on COLAB, we can finally run but did not completely retrain the DGCRN model, luckily due to the saved/pre-trained weights in its GitHub repository.

Table 4: Experiment results of the state-of-the-art models (the ECE episode)

		Key Metrics		Other Metrics	
Architecture	**MSE**	**RMSE**	**MAE**	**Type**	**Value**
Taxi-Simple-LSTM-Pytorch	72269860	8501.168	5753.569		
Uber-Simple-LSTM-Pytorch	0.0332864	0.1824457	0.13018544		
Taxi-Simple-LSTM-Keras	93431556	9666	92.484945		
Uber-Simple-LSTM-Keras	105050200.4	10249.4	83.21		
CRANN-Temporal	6653.6636	81.569992	50.3727684	Bias	-3.82883501
				Relative error %	24.85518008
CRANN-Spatial	55740.719	236.09472	117.7406616	Bias	-14.2317371
				Relative error %	9.053748846
CRANN-Dense	67264.055	259.35315	138.2879333	Bias	-4.71734381
				Relative error %	24.85518008
Seq2seq (flow) [14]	1626.986623	40.33592224	21.3		
GAT Seq2seq (flow) [14]	1098.041371	33.13670731	18.3		
ST-Metanet (flow) [14]	813.1885148	28.51646042	16.9		
Seq2seq (speed) [14]	44.4751941	6.668972492	3.55		
GAT Seq2seq (speed) [14]	36.92343427	6.076465607	3.28		
ST-Metanet (speed) [14]	33.63158961	5.799274921	3.05		
AGCRN - PeMSD4	1040.761367	32.26083333	19.69416667	MAPE (TWCC)	13.02040833
AGCRN - PeMSD8	718.9101563	26.8125	17.01583333	MAPE(COLAB)	10.65138333
ASTGCN - PeMSD4	1173.7476	34.26	21.84	MAPE	0.15
ASTGCN - PeMSD8	809.4025	28.45	18.5	MAPE	0.11
Deepforecast	2.481881285	1.5753988	1.1590073	NRMSE_maxmin(%)	15.575216
				NRMSE_mean(%)	43.097104
DCRNN(Metr-LA)-STA - 15min	75.5161	8.69	4.02	MAPE	9.39
DCRNN(Metr-LA)-STA - 1hr	201.9241	14.21	6.79	MAPE	16.71
DCRNN(Metr-LA)-VAR- 15min	60.5284	7.78	4.37	MAPE	10.08
DCRNN(Metr-LA)-VAR - 1hr	114.0624	10.68	6.5	MAPE	15.84
DCRNN(Pemsbay)-STA - 15min	11.7649	3.43	1.6	MAPE	3.24
DCRNN(Pemsbay)-STA - 1hrmin	49.2804	7.02	3.05	MAPE	6.84
DCRNN(Pemsbay)-VAR - 15min	9.5481	3.09	1.74	MAPE	3.59
DCRNN(Pemsbay)-VAR - 1hr	26.1121	5.11	2.92	MAPE	6.46
STGCN - 15min	16.532356	4.066	2.231	%Wmape	5.206
STGCN - 30 min	33.051001	5.749	3.04	%Wmape	7.386
STGCN - 45min	46.744569	6.837	3.623	%Wmape	8.927
Spacetimeformer [80]	36.21	6.017474553	2.82	MAPE	7.71
				model.loss	0.258

For the ST-METANET model, 64GB+ RAM is a safe level - as we cannot afford to rent a suitable cloud VM for it, we decided to put the model as a basis, directly referring to the original results in pink cells – the same measure also applies to Spacetimeformer owing to the same error occurring to the GPU.

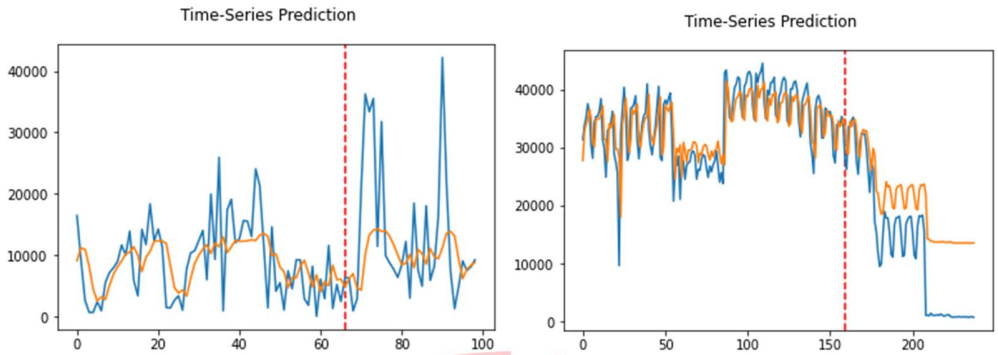


Figure 33: Pytorch-LSTM prediction results for (left) UBER and (right) TAXI ridership data

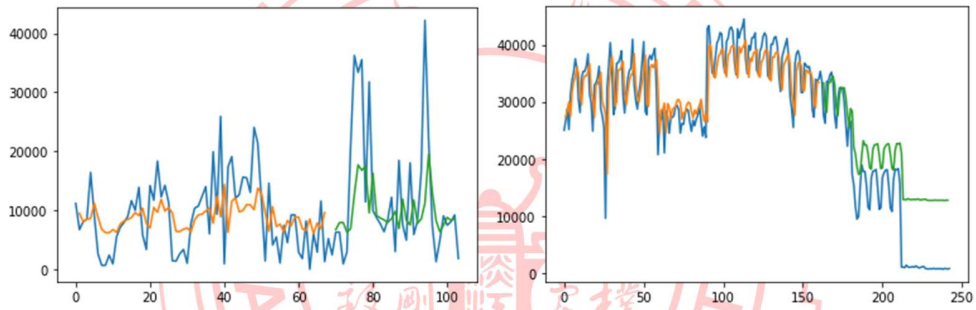


Figure 34: Keras-LSTM prediction results for (left) UBER and (right) TAXI ridership data

To explain Table 4, the simple LSTM models had a tremendous RMSE error in thousands due to the model's simplicity, overfitting, and gradient explosion. The UBER-LSTM Pytorch model is somewhat accurate because the Pytorch-LSTM model is perfectly hyperparameter-tuned for time-series with apparent seasonality. Nevertheless, we should never trust the numbers unless we see the graphs of Figures 33 and 34. The CRANN Model, despite setting to 200-epoch maximum, automatically finished training around 100 epochs. Using TWCC instance with 2x GPUs, after 2 hours of training, the AGCRN produced numerically the same results as its paper - however, for the same model run on a COLAB VM, if able to, the results are more divergent from the reference due to its inferior nature compared to paid VMs.

It is also evident that when learning correct datasets, complex state-of-the-art models (such as CRANN) vastly outperformed simple models (like Keras-LSTM). As we can see in Table 4, the "Deepforecast" Multi-LSTM model, to date, is the best model for spatiotemporal traffic prediction due to the depth of a classic but straightforward to calibrate LSTM – nothing could match it when trained by MS-Winds dataset. Nevertheless, it takes 2 hours to train in COLAB. For the same model, PEMS-BAY dataset trains it better than the METR-LA due to shear larger data size.

A rivalling model to Deepforecast is DCRNN trained by PEMS-Bay dataset, with both static (STA) and variable (VAR) modes, which has a comparable performance; like 1.16 vs 1.6(STA) or 1.74(VAR) MAE, and 1.58 vs 3.43(STA) 3.09(VAR) RMSE

4.2. Experiment on our Custom Dataset. (a.k.a. Custom Event Heatmap Experiment; CEHE)

4.2.1 CEHE Experiment Settings & Details

In this (2nd) episode, we have an event dataset (with coordinates & timestamps) of fire incident record data of New Taipei City, Taiwan, and we will choose referenced models. To start, we selected all models in section 2.2, trained them with the NTPC fire data, and ran it under the Google-COLAB environment. Each ".ipynb" notebook contains the data input and processing section. We started with the dataset having its dots, recording both timestamps and coordinates, extracted. Then, the "cytogenesis" function, fed with the extracted data and a square map width, converted dots to raster heatmap, producing 36 images, each representing each month in the 2015-17 timeframe. The testing heatmaps belonged to March, June, September, and December, while the training data belonged to other months. For the temporal domain, however, each timestamp is extracted and has a frequency of 1 day – when each timestamp is grouped in a timeframe, be it daily, weekly, or monthly, the frequency also counts against the timeframe, obtaining a time-series recording incident frequencies by timeframe. Only the daily series is used for prediction in our time-domain models unless specified by other timeframe widths. All models had the Adam optimizer with a learning rate of 0.001 and were trained for 100 epochs. Table 5 shows its preliminary results

Table 5: Models and results in the NTPC fire CEHE (both domains)

MODEL	DOMAIN	Batch size / frequency	Train Metrics:				Test Metrics:			
			MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
Pre-MMS-LSTM4	Time		91.769263	9.579627	5.283182	0.248342	52.195229	7.224627	5.508191	0.37734
Post-MMS-LSTM4-noLB	Time	1day +1day-lookback	7.679134	2.771125	1.556069	0.133908	61.438013	7.83824	5.954068	0.29369
Post-MMS-LSTM-LB5	Time	1day +5day-lookback	7.582465	2.753628	1.559106	0.138684	57.692435	7.595554	5.70559	0.283949
BAE	Space	24 train / 12 test / 9 show	4.636669	2.1532927	1.8429354	1541306600	5.944328	2.4380991	2.1663473	1860365700
DCGAN	Space	12	8.476757	2.91148709	0.80054784	38606776	7.6664495	2.76883541	0.7418378	38433640
Fbprophet (37 days)	Time	1 day	n/a	n/a	n/a	n/a	48.613108	6.97231	5.045342	0.281977
CVAE	Space	9	7.6662908	2.7688065	0.3827354	177096430	6.7336392	2.5949256	0.3442523	174446380
VAE	Space	9	1.3036826	1.14178919	0.37329638	4.11115E+14	1.0857255	1.04198155	0.3522904	4.2458E+14
WGAN-GP	Space	9	12.2007065	3.49295097	0.8706569	8.83334E+12	10.965601	3.31143488	0.81602687	8.84299E+12
Weekly ARIMA	Time	7 days	n/a	n/a	n/a	n/a	1436.90057	37.9064714	28.42712296	0.195063926
LSGAN	Space	12	197.77995	14.0634261	7.245772	2.64E+16	194.74649	13.95516	7.1890564	2.63E+16
Post-MMS-LSTM-Weekly	Time	7 days	227.370864	15.078822	10.250484	0.086445	852.383545	29.195608	21.007481	0.143819
Post-MMS-LSTM9	Time	1day +5day-lookback	7.578619	2.752929	1.563434	0.139068	55.893509	5.60772	5.70559	0.281344

In Table 5, the green cells indicate the best available model: the red cells, the worst. The purple cells indicate rivalling metrics. In one of our hyperparameter subtests, we increased the batch size to 12 for all Spatial models, except for the BAE. Consider viewing Table 5 for the results.

2.2 CEHE Results

Supposedly, more complex and advanced models usually outperform the simple, basic ones. As shown in Table 5, Table 6, and Figure 35, for the autoencoders, due to simplicity, and well-tuned hyperparameters despite minimal touches, of VAE over CVAE when trained by our tiny dataset, the VAE outperformed the CVAE. It is ironic that convolution in the VAE, i.e., CVAE, does not always produce better MSE (but still better MAE) than the VAE. Nevertheless, both autoencoders outperformed their basic counterparts (BAE). Usually, GANs often outperform Autoencoders, but it is not always true in our case due to our small dataset size. Some exceptions are DCGAN & WGAN GP out-MAE'd the BAE. Interestingly, LSGAN is much harder to train than WGAN and DCGAN proven by its notably poor performance when trained by our small dataset of rasterised heatmaps as it did not finish the training due to the architecture designed for a low learning rate. In Table 5 of the time domain, try to notice hidden hyperparameter subtests, which is to be further analysed in section 4.4.

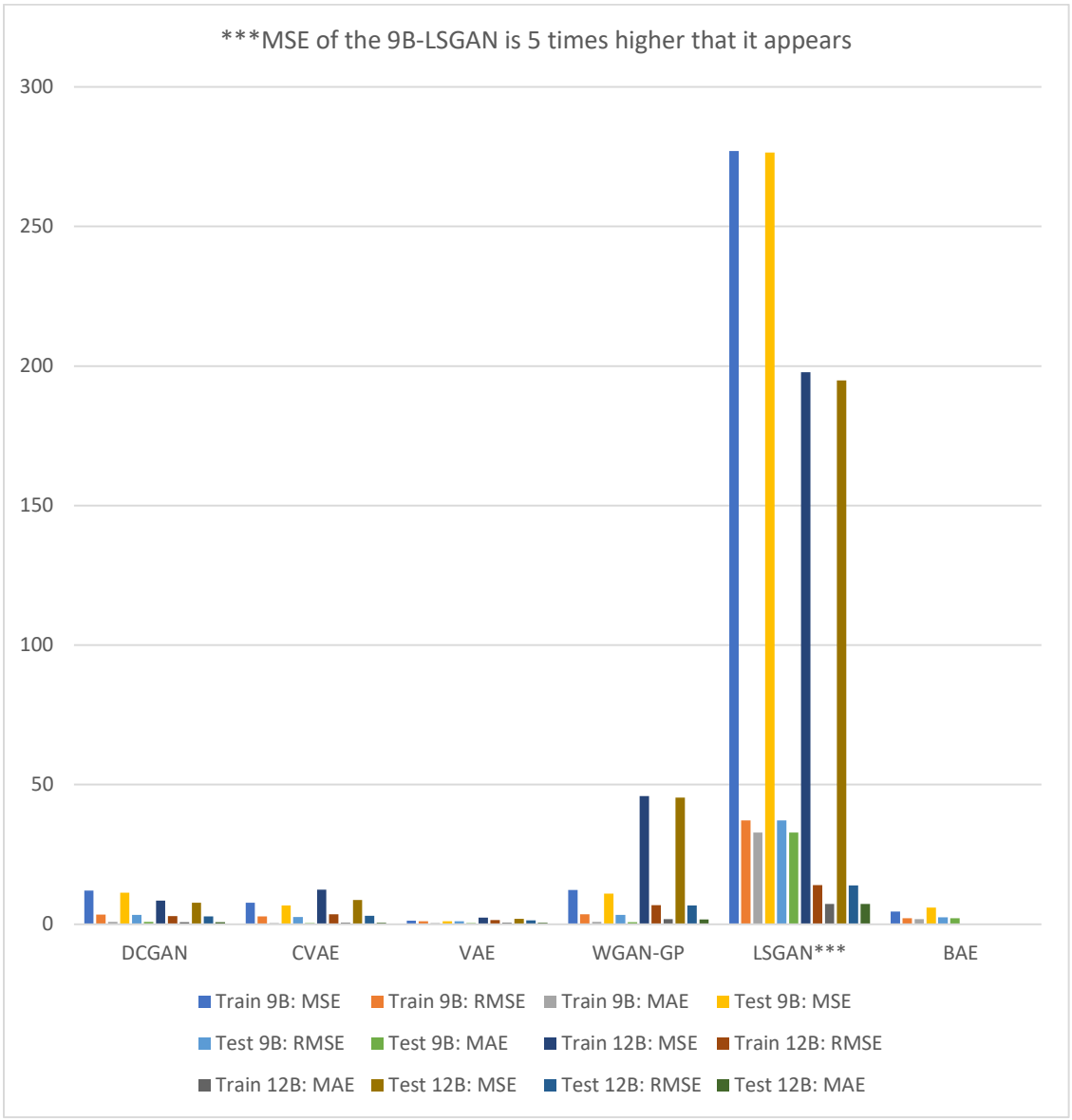


Figure 35: Metrics of CEHE Spatial Models

For the Temporal domain, it is usual that FBProphet and LSTM outperform ARIMA, even with the same frequency. Notably, for our small non-seasonal time-series, FBProphet outperformed the LSTMs because this architecture did not overfit; and never overfits. Furthermore, we have observed that the PreMMS-LSTM, not yet overfit, can still be trained and hyperparameter-tuned to overtake the FBProphet model. With overfitting occurring on the PostMMS-LSTM, it became so saturated that it could not be trained further. Nevertheless, we carried out hidden hyperparameter subtests with it. Furthermore, adding some lookback steps to the LSTM also helped with training, improving test metrics. Our future suggestion is to prioritise the PreMMS-LSTM and do some hyperparameter subtests with it.

Table 6: Models and results in the CEHE: the spatial domain with batch sizes of 9 and 12

		DCGAN	CVAE	VAE	WGAN-GP	LSGAN***	BAE
Train 9B:	MSE	12.011371	7.6662908	1.3036826	12.2007065	1385.0496	4.636669
	RMSE	3.46574244	2.7688065	1.14178919	3.49295097	37.21625398	2.1532927
	MAE	0.8862971	0.3827354	0.37329638	0.8706569	32.86514	1.8429354
Test 9B:	MSE	11.25666	6.7336392	1.0857255	10.965601	1382.266	5.944328
	RMSE	3.3550947	2.5949256	1.04198155	3.31143488	37.17883795	2.4380991
	MAE	0.8440172	0.3442523	0.3522904	0.81602687	32.839905	2.1663473
Train 12B:	MSE	8.476757	12.374425	2.326788	45.895508	197.77995	4720.15986
	RMSE	2.91148709	3.5177302	1.52538125	6.77462234	14.06342608	95.3192812
	MAE	0.80054784	0.48861146	0.5209149	1.813967	7.245772	0
Test 12B:	MSE	7.6664495	8.672948	1.9275316	45.41533	194.74649	0
	RMSE	2.76883541	2.9449868	1.38835572	6.73908963	13.95515999	0
	MAE	0.7418378	0.41251573	0.48643875	1.7555918	7.1890564	0

To describe how the Pre- and the Post- MMS LSTM are different, according to the source code of the LSTM's pre-entry data preparation block, the Pre-MMS runs the MMS process before the train/test split (TTS), obtaining an obviously better test (but worse train) metrics of the same LSTM. The PostMMS, on the other hand, runs the MMS after the TTS. achieving much better train (but slightly worse) metrics.

To date, the FBProphet model in the temporal domain and the VAE in the spatial domain are our best models trained with our NTPC-fire dataset. For visualized results, Figure 36 denotes all rasterized heatmaps generated by each spatial model, and Figure 37 displays the temporal prediction results.

According to Table 5, Table 6 and Figure 35, upon adding more images to the training batch, , the DCGAN and the LSGAN improved, and the others worsened. The CVAE and the VAE rivalled the MAE values, with CVAE producing better MAE most of the time. The BAE did not enter this subtest. Nevertheless, the VAE is still the best model for our dataset. Only for PostMMS-LSTMs, graphs were plotted from June 1, 2017, onwards. Visualised predictions of Table 5 can be viewed in Figures 36, 37 and 38.

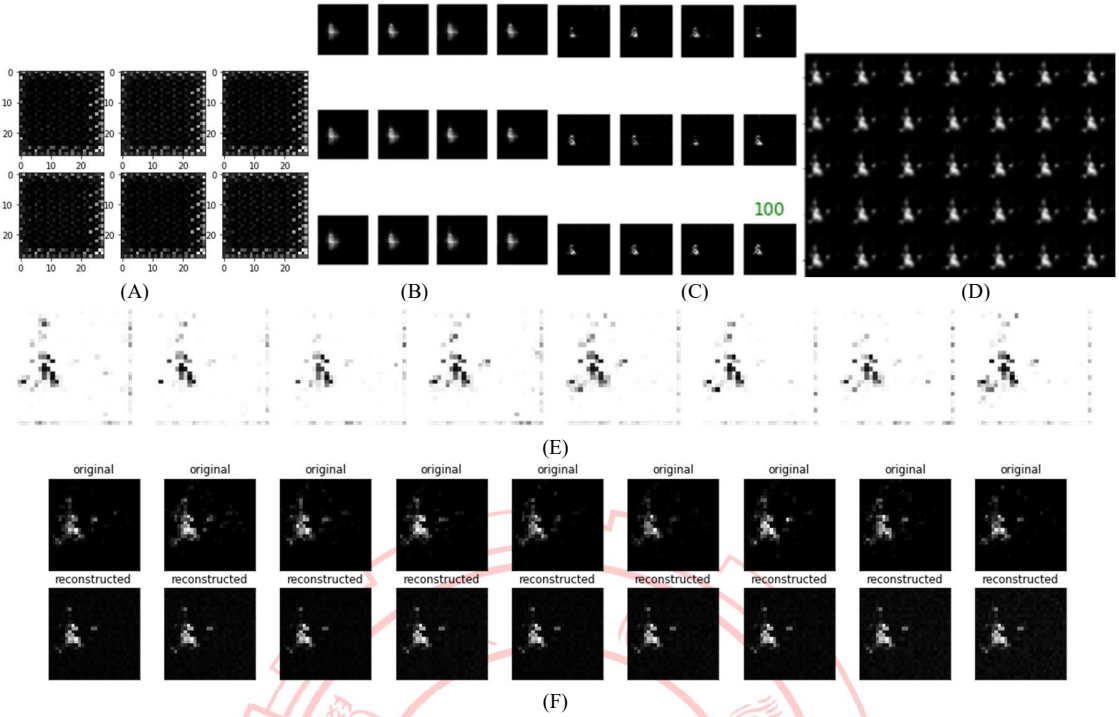


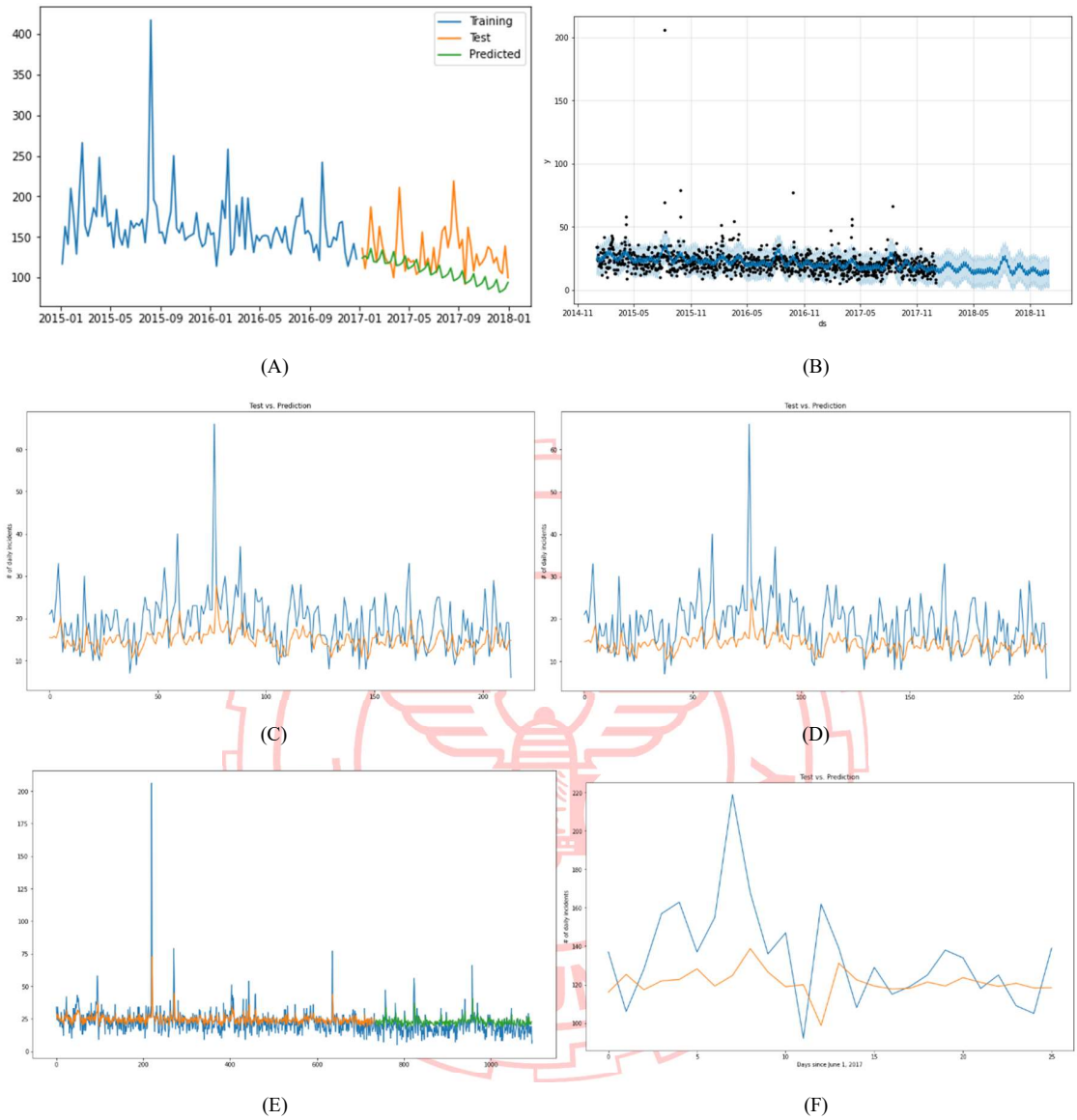
Figure 36: The visualised NTPC-Fire raster heatmap prediction generated by some different models: (A) = LSGAN, (B)=CVAE, and (C)=DCGAN. (D)=VAE, (E)=WGAN-GP, (F)=BAE with the input on the top. After 100 epochs, we can see that VAE and BAE were not only easy to train but also produced realistic reconstructions.

4.3. Accomplished Tasks

So far, we have read more than 80 reference papers to study their models and ran the following in the first episode (ECE): 1. Taxi Uber LSTMs, ST-METANET, CRANN, AGCRN, ASTGCN, DCRNN, STGRN, Deepforecast Multi-LSTM, and Spacetimeformer. With some models out of the scope and possessing incompatible metrics, the Lotto-Att-LSTM and TPA-LSTM had their results cancelled. With excessive troubleshooting and debugging, either in COLAB or TWCC, most models were able to run successfully and produced results - only two models failed to acquire their suitable environment, namely ST-METANET and Spacetimeformer.

The TAXI-Uber Dataset, in the future, is to be done with the spatial domain. The outline and the technical basics are written in the first section. If you are interested more in STDM, you may read their references.

In our second episode (CEHE), for the spatial domain, we ran Autoencoders (BAE, VAE, CVAE) and GANs (DCGAN, LSGAN, WGAN-GP) to reconstruct rasterised heatmaps. Meanwhile, for the temporal domain, we ran FBProphet, Auto-ARIMA, and LSTMs with different configures for hidden subtests. Results, visualisation, and explanation for both episodes of the experiment are added to sections 4.1 and 4.2, with further analysis, explanation, and future suggestions described in section 4.4.



(A) (B)

(C) (D)

(E) (F)

Figure 37: The visualised time-series prediction of some different models: (A)= ARIMA, (B)=FBProphet, (C)=PostMMS-LSTM4 with 5-day lookback. (D)=PostMMS-LSTM4 with no lookback, notice the lower orange line compared to subfigure(C). (E) PreMMS LSTM, not yet overfit. (F) PostMMS-LSTM-Weekly. All of them predicted a decreasing trend of incident frequency.

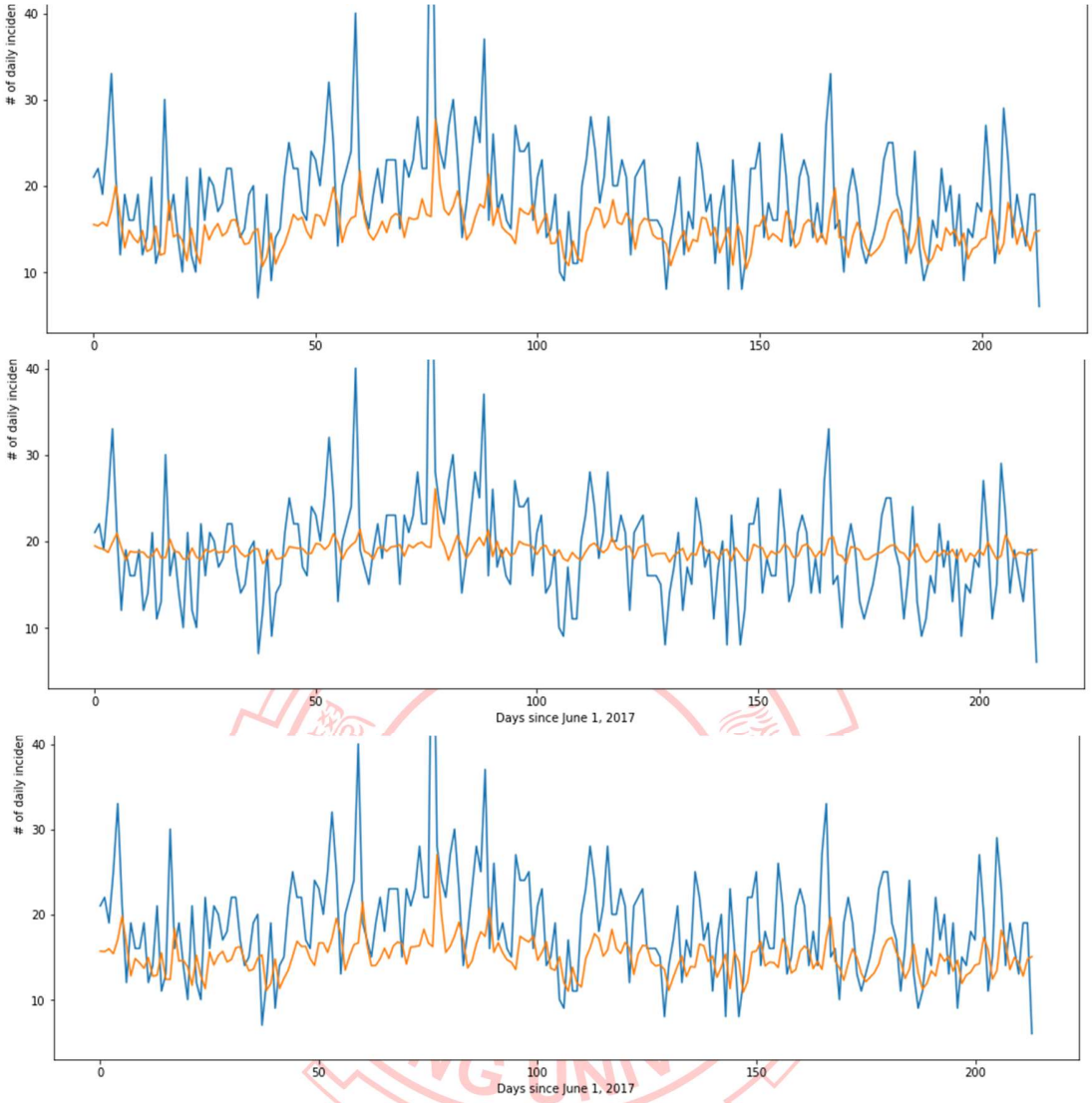


Figure 38: The PostMMS-LSTM's prediction of different configurations: (Top): Original with 4-LSTM cells, (Middle): Trained with weekly series and tested against the daily series, notice the flat graph, the worst predictions of all three. (Bottom): with 9 cells in the LSTM layer, notice the smoother troughs and higher tips. To sum up, hyperparameter tuning on simple LSTM did not evidently affect the prediction unless the training data changed.

4.4 Discussion

Restating the results in section 4.1.2, the first experiment, the ECE, arranged a competition between some notable state-of-the-art traffic forecasting models to date and discovered that, when trained with the same or similar dataset, the newer, more architecturally complex models do not always outperform the older ones. Nevertheless, the Deepforecast multi-LSTM model overtook everything on our board, proven by the lowest values of the three metrics. Besides, the ST-Metanet shares similar metrics with DRCNN-Pemsbay, STGCN, and Spacetimeformer.

For the second experiment's (the CEHE on a small NTPCfire dataset) spatial domain, the GANs, while more architecturally advanced than Autoencoders, did not outperform them (except

the simplest BAE); due to GANs, compared to the AEs, requiring a large dataset, and being harder to train and hyperparameter-tune, meaning that Autoencoders suited smaller datasets (24 training images, 36 total images) better than GANs, because of the ease of training.

Furthermore, in the spatial CEHE apparent subtest (Table 5), adjusting batch size also affect the training state. For instance, increasing the batch size by 3 (9 -> 12) improved some models while worsening some. Thus, choosing a batch size must be based on result optimisation, which is to be done in the future. In the future, we recommend that you plan another subtest varying even more batch sizes against metrics and choose the best one producing the least errors while controlling everything else.

Usually, in the temporal domain, the LSTM greatly outperforms ARIMA but only slightly outperforms FBProphet. However, there is an exception; for time-series with no apparent seasonality, like our fire data, FBProphet outperformed the LSTM because of the FBProphet small data size requirement compared to the LSTM. Nevertheless, the PreMMS-LSTM, if well trained and tuned, could possibly defeat the FBProphet. We recommend you perform hyperparameter subtests with this model. Generally speaking, LSTM, while being a generalist, require a sizeable time series, usually with apparent seasonality, to train appropriately and maximise its efficiency. Evidently, FBProphet is even more generalist because, unlike the LTSM, it better suits small datasets as ARIMA does. Honestly, in the future, we would like to recommend you do even more subtests; like number of LSTM cells, train/test ratio, etc.

Noticing the hidden hyperparameter subtest in Table 5, LSTM, a neural network, is more prone to overfitting and gradient diminish & explosion. Adding a lookback (1 -> 5), as well as adding more LSTM cells (4 -> 9), also helps with training, slightly improving metrics. Negatively, in an LSTM, extending time-series frequency hampers its learning ability. Overall, in the CEHE, the FBProphet and the VAE are our best models to be implemented on the dot-to-raster heatmap event data. The hidden subtests will eventually lead to a model design approach called “hyperparameter tuning”. In the future, you may add more LSTM, AE, and GAN variants to our CEHE.

CHAPTER 5: CONCLUSIONS

This paper conducted a comprehensive overview of recent advances in machine and deep learning techniques for STDM. We first categorised the different data types and representations of ST data and briefly introduced the popular deep learning models used for STDM. Next, we focused on some STDM techniques using deep neural networks. Then, we reviewed recent works based on the STDM tasks, including prediction, clustering, anomaly detection, estimation and inference, and others. Next, we performed a comparative experiment of traffic prediction architectures (our ECE) to see which one produced the tiniest error in terms of MAE, MSE and RMSE, concluding that each model did well for their respective architectures and discovering two facts: One, the newer, more architecturally advanced models do not always outperform the older, simpler ones. And two, the Deepforecast Multi-LSTM [73], to date, performed the best STDM learning and prediction. For a non-graph event data prediction, we compared each popular model to generate rasterised heatmaps and perform time-series forecasting, resulting in a discovery that the VAE in the spatial domain and the FBProphet in the temporal domain did the best at their respective jobs. Also, to study how the STDM process works for a custom dataset, we chose some suitable models for each domain for generation & forecasting tasks - and compared their performance. The hyperparameter subtests in our CEHE can eventually lead to hyperparameter tuning, a model improvement measure. Finally, we listed some open problems, provided recommendations for future subtests, discussed, and pointed out the current issues, especially ones of the runtime environment, and indicated the future research directions for this fast-growing research field.

REFERENCES

[1] Doshi, J., Basu, S. and Pang, G., 2018. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*.

[2] Doshi, Jigar. 2018. Residual Inception Skip Network for Binary Segmentation. Pages 216–219 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

[3] Amit, S.N.K.B. and Aoki, Y., 2017, September. Disaster detection from aerial imagery with convolutional neural network. In 2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC) (pp. 239-245). IEEE.

[4] V. Iglovikov, S. Mushinskiy, and V. Osin, "Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition," vol. June 2017.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," ArXiv, vol. abs/2004.10934, 2020.

[6] Fang, Z., Wang, Y., Peng, L. and Hong, H., 2020. Predicting flood susceptibility using LSTM neural networks. *Journal of Hydrology*, [online] p.125734. Available at: <https://doi.org/10.1016/j.jhydrol.2020.125734> [Accessed 18 April 2021].

[7] Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

[8] S. Wang, J. Cao, & P. Yu, "Deep Learning for Spatio-Temporal Data Mining: A Survey," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.3025580.

[9] Yu, B., Yin, H., & Zhu, Z. (2017). Spatiotemporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

[10] Correa, D., Xie, K., & Ozbay, K. (2017). Exploring the taxi and Uber demand in New York City: An empirical analysis and spatial modeling. In *96th Annual Meeting of the Transportation Research Board, Washington, DC*.

[11] Amato, F., Guignard, F., Robert, S., & Kanevski, M. (2020). A novel framework for spatiotemporal prediction of environmental data using deep learning. Scientific Reports, 2020(10), 22243. doi: 10.1038/s41598-020-79148-7

[12] Tang, Q., Yang, M., & Yang, Y. (2019). ST-LSTM: A Deep Learning Approach Combined Spatiotemporal Features for Short-Term Forecast in Rail Transit. Journal of Advanced Transportation, 2019, Article ID 8392592. doi:https://doi.org/10.1155/2019/8392592

[13] Lu, H., Huang, D., Song, Y., Jiang, D., Zhou, T., & Qin, J. (2020). ST-TrafficNet: A Spatial-Temporal Deep Learning Network for Traffic Forecasting. Electronics, 2020(9), 1474.

[14] Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., & Zhang, J. (2019, July 25). Urban Traffic Prediction from Spatiotemporal Data Using Deep Meta Learning. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/3292500.3330884

[15] de Medrano, R., & Aznarte, J. L. (2020). A spatiotemporal attention-based spot-forecasting framework for urban traffic prediction. *Applied Soft Computing*, *96*, 106615.

[16] Luo, X., Li, D., & Zhang, S. (2019). Traffic flow prediction during the holidays based on DFT and SVR. *Journal of Sensors*, *2019*.

[17] Jiawei Han, Micheline Kamber, Jian Pei, 13 - Data Mining Trends and Research Frontiers, Morgan Kaufmann, 2012, Pages 585-631.

[18] Shih, S. Y., Sun, F. K., & Lee, H. Y. (2019). Temporal pattern attention for multivariate time-series forecasting. *Machine Learning*, *108*(8), 1421-1441.

[19] Atluri, G., Karpatne, A., & Kumar, V. (2018). Spatiotemporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)*, *51*(4), 1-41.

[20] Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, *60*(1), 208-221.

[21] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

[22] Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, *14*(5), 1003-1016.

[23] Chen, X., Faghmous, J. H., Khandelwal, A., & Kumar, V. (2015, June). Clustering dynamic spatiotemporal patterns in the presence of noise and missing data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[24] Chen, M., Davis, J. M., Liu, C., Sun, Z., Zempila, M. M., & Gao, W. (2017, September). Using deep recurrent neural network for direct beam solar irradiance cloud screening. In *Remote Sensing and Modeling of Ecosystems for Sustainability XIV* (Vol. 10405, p. 1040503). International Society for Optics and Photonics.

[25] Cheng, L., Zang, H., Ding, T., Sun, R., Wang, M., Wei, Z., & Sun, G. (2018). Ensemble recurrent neural network based probabilistic wind speed forecasting approach. Energies, 11(8), 1958.

[26] Hossain, M., Rekabdar, B., Louis, S. J., & Dascalu, S. (2015, July). Forecasting the weather of Nevada: A deep learning approach. In 2015 international joint conference on neural networks (IJCNN) (pp. 1-6). IEEE.

[27] Liu, H., Mi, X., & Li, Y. (2018). Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM. Energy Conversion and Management, 159, 54-64.

[28] Rasp, S., & Lerch, S. (2018). Neural networks for postprocessing ensemble weather forecasts. Monthly Weather Review, 146(11), 3885-3900.

[29] Zaytar, M. A., & El Amrani, C. (2016). Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. International Journal of Computer Applications, 143(11), 7-11.

[30] Ke, J., Yang, H., Zheng, H., Chen, X., Jia, Y., Gong, P., & Ye, J. (2018). Hexagon-based convolutional neural network for supply-demand forecasting of ride-sourcing services. IEEE Transactions on Intelligent Transportation Systems, 20(11), 4160-4173.

[31] Lee, D., Jung, S., Cheon, Y., Kim, D., & You, S. (2018). Forecasting taxi demands with fully convolutional networks and temporal guided embedding.

[32] Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016, October). DNN-based prediction model for spatiotemporal data. In Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems (pp. 1-4).

[33] Zhu, Q., Chen, J., Zhu, L., Duan, X., & Liu, Y. (2018). Wind speed prediction with spatio–temporal correlation: A deep learning approach. Energies, 11(4), 705.

[34] Lv, J., Li, Q., Sun, Q., & Wang, X. (2018, January). T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction. In 2018 IEEE international conference on big data and smart computing (bigcomp) (pp. 82-89). IEEE.

[35] Song, X., Kanasugi, H., & Shibasaki, R. (2016, July). Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (pp. 2618-2624).

[36] Chattopadhyay, A., Hassanzadeh, P., & Pasha, S. (2018). A test case for application of convolutional neural networks to spatiotemporal climate data: Re-identifying clustered weather patterns. arXiv preprint arXiv:1811.04817.

[37] Nguyen, N. T., Wang, Y., Li, H., Liu, X., & Han, Z. (2012, December). Extracting typical users' moving patterns using deep learning. In 2012 IEEE Global Communications Conference (GLOBECOM) (pp. 5410-5414). IEEE.

[38] Zhang, J., Zheng, Y., & Qi, D. (2017, February). Deep spatiotemporal residual networks for citywide crowd flows prediction. In Thirty-first AAAI conference on artificial intelligence.

[39] Gardner Jr, E. S. (2006). Exponential smoothing: The state of the art—Part II. International journal of forecasting, 22(4), 637-666.

[40] Box, G. E., & Jenkins, G. M. (1976). Time-series Analysis. Forecasting and Control (rev. ed.).

[41] Aoki, M. (2013). State space modeling of time-series. Springer Science & Business Media.

[42] Kelejian, H. H., & Prucha, I. R. (1999). A generalized moments estimator for the autoregressive parameter in a spatial model. International economic review, 40(2), 509-533.

[43] Brunsdon, C., Fotheringham, S., & Charlton, M. (1998). Geographically weighted regression. Journal of the Royal Statistical Society: Series D (The Statistician), 47(3), 431-443.

[44] Oliver, M. A., & Webster, R. (1990). Kriging: a method of interpolation for geographical information systems. International Journal of Geographical Information System, 4(3), 313-332.

[45] Kasetkasem, T., & Varshney, P. K. (2002). An image change detection algorithm based on Markov random field models. IEEE Transactions on Geoscience and Remote Sensing, 40(8), 1815-1823.

[46] Schroder, M., Rehrauer, H., Seidel, K., & Datcu, M. (1998). Spatial information retrieval from remote-sensing images. II. Gibbs-Markov random fields. IEEE Transactions on geoscience and remote sensing, 36(5), 1446-1455.

[47] Zhao, Y., Zhang, L., Li, P., & Huang, B. (2007). Classification of high spatial resolution imagery using improved Gaussian Markov random-field-based texture features. IEEE Transactions on Geoscience and Remote Sensing, 45(5), 1458-1468.

[48] Zhang, P., Jia, Y., Gao, J., Song, W., & Leung, H. (2018). Short-term rainfall forecasting using multi-layer perceptron. IEEE Transactions on Big Data, 6(1), 93-106.

[49] Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., & Li, T. (2018). Predicting citywide crowd flows using deep spatiotemporal residual networks. Artificial Intelligence, 259, 147-166.

[50] Zhao, P., Zhu, H., Liu, Y., Li, Z., Xu, J., & Sheng, V. S. (2018). Where to go next: A spatiotemporal LSTM model for next POI recommendation. arXiv preprint arXiv:1806.06671.

[51] Duan, L., Hu, T., Cheng, E., Zhu, J., & Gao, C. (2017). Deep convolutional neural networks for spatiotemporal crime prediction. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE) (pp. 61-67). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[52] Ding, D., Zhang, M., Pan, X., Wu, D., & Pu, P. (2018, April). Geographical feature extraction for entities in location-based social networks. In Proceedings of the 2018 World Wide Web Conference (pp. 833-842).

[53] Costilla-Reyes, O., Scully, P., & Ozanyan, K. B. (2017). Deep neural networks for learning spatiotemporal features from tomography sensors. IEEE Transactions on Industrial Electronics, 65(1), 645-653.

[54] Wen, D., Wei, Z., Zhou, Y., Li, G., Zhang, X., & Han, W. (2018). Deep learning methods to process fmri data and their application in the diagnosis of cognitive impairment: a brief overview and our opinion. Frontiers in neuroinformatics, 12, 23.

[55] Dvornek, N. C., Ventola, P., Pelphrey, K. A., & Duncan, J. S. (2017, September). Identifying autism from resting-state fMRI using long short-term memory networks. In International Workshop on Machine Learning in Medical Imaging (pp. 362-370). Springer, Cham.

[56] Guo, X., Dominick, K. C., Minai, A. A., Li, H., Erickson, C. A., & Lu, L. J. (2017). Diagnosing autism spectrum disorder from brain resting-state functional connectivity patterns using a deep neural network with a novel feature selection method. Frontiers in neuroscience, 11, 460.

[57] Heinsfeld, A. S., Franco, A. R., Craddock, R. C., Buchweitz, A., & Meneguzzi, F. (2018). Identification of autism spectrum disorder using deep learning and the ABIDE dataset. NeuroImage: Clinical, 17, 16-23.

[58] Horikawa, T., & Kamitani, Y. (2017). Generic decoding of seen and imagined objects using hierarchical visual features. Nature communications, 8(1), 1-15.

[59] Kim, J., Calhoun, V. D., Shim, E., & Lee, J. H. (2016). Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification

performance: Evidence from whole-brain resting-state functional connectivity patterns of schizophrenia. Neuroimage, 124, 127-146.

[60] Meszlényi, R. J., Buza, K., & Vidnyánszky, Z. (2017). Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture. Frontiers in neuroinformatics, 11, 61.

[61] Shi, J., Zheng, X., Li, Y., Zhang, Q., & Ying, S. (2017). Multimodal neuroimaging feature learning with multimodal stacked deep polynomial networks for diagnosis of Alzheimer's disease. IEEE journal of biomedical and health informatics, 22(1), 173-183.

[62] Cheng, W., Shen, Y., Zhu, Y., & Huang, L. (2018, April). A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In Thirty-second AAAI conference on artificial intelligence.

[63] Zhang, H., Wu, H., Sun, W., & Zheng, B. (2018). Deeptravel: a neural network-based travel time estimation model with auxiliary supervision. arXiv preprint arXiv:1802.02147.

[64] Kut, A., & Birant, D. (2006). Spatiotemporal outlier detection in large databases. Journal of computing and information technology, 14(4), 291-297.

[65] Cheng, T., & Li, Z. (2004, June). A hybrid approach to detect spatial-temporal outliers. In Proceedings of the 12th International Conference on Geoinformatics Geospatial Information Research (pp. 173-178).

[66] Cheng, T., & Li, Z. (2006). A multiscale approach for spatio-temporal outlier detection. Transactions in GIS, 10(2), 253-263.

[67] Zhang, Z., He, Q., Gao, J., & Ni, M. (2018). A deep learning approach for detecting traffic accidents from social media data. Transportation research part C: emerging technologies, 86, 580-596.

[68] Zhu, L., Guo, F., Krishnan, R., & Polak, J. W. (2018, November). A deep learning approach for traffic incident detection in urban networks. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 1011-1016). IEEE.

[69] Racah, E., Beckham, C., Maharaj, T., Kahou, S. E., & Pal, C. (2016). ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. arXiv preprint arXiv:1612.02095.

[70] Wang, J., Wu, N., Zhao, W. X., Peng, F., & Lin, X. (2019, July). Empowering A* search algorithms with neural networks for personalized route recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 539-547).

[71] Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *arXiv preprint arXiv:2007.02842*.

[72] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019, July). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 922-929).

[73] Ghaderi, A., Sanandaji, B. M., & Ghaderi, F. (2017). Deep forecast: Deep learning-based spatiotemporal forecasting. *arXiv preprint arXiv:1707.08110*.

[74] F. Chollet, “Keras documentation: Variational Autoencoder,” Keras, 01-Apr-2020. [Online]. Available: https://keras.io/examples/generative/. [Accessed: 19-Dec-2021].

[75] A. of the Google Dev Team, “Introduction | generative adversarial networks | google developers,” Generative Adversarial Networks, 08-Oct-2019. [Online]. Available: https://developers.google.com/machine-learning/gan. [Accessed: 20-Dec-2021].

[76] M. G. Heberger, “Artificial Neural Network,” Wikipedia, 02-Dec-2021. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed: 20-Dec-2021].

[77] C. Duong, “Facebook Prophet- Forecasting at scale.,” Facebook Prophet, 14-Jul-2018. [Online]. Available: https://facebook.github.io/prophet/. [Accessed: 20-Dec-2021].

[78] B. Lamberta, “Github: TensorFlow Generative Models Tutorial,” GitHub, 13-Mar-2021. [Online]. Available: https://github.com/tensorflow/docs/tree/master/site/en/tutorials/generative. [Accessed: 21-Dec-2021].

[79] B. Lamberta, “Deep convolutional generative Adversarial Network: TensorFlow Core,” TensorFlow, 13-Mar-2021. [Online]. Available: https://www.tensorflow.org/tutorials/generative/dcgan. [Accessed: 21-Dec-2021].

[80] Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-Range Transformers for Dynamic Spatiotemporal Forecasting. arXiv preprint arXiv:2109.12218.

[81] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028.

[82] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2794-2802).

[83] S. Guo, Y. Lin, H. Wan, X. Li and G. Cong, "Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2021.3056502.

[84] J. Brownlee, “Machine learning mastery,” Machine Learning Mastery, 25-Oct-2021. [Online]. Available: https://machinelearningmastery.com/. [Accessed: 13-Jan-2022].

APPENDIX:
The Journal Version of this Thesis

Applications of Deep Learning in GIS: Spatiotemporal data mining and forecasting

Yijia Tsai
iaiclab.tku@gmail.com

Supasin Wuthikulphakdi
608785068@gms.tku.edu.tw

iAIC Lab, Department of Computer Science and Information Engineering
Tamkang University
151 Injuang Road, Tamsui, New Taipei, Taiwan, R.O.C.

ABSTRACT

This paper addresses two issues: 1. Among those recently proposed STDM-DL (spatial-temporal data mining, deep learning) models, how to compare those models in forecasting capability? And 2. What are the performances of those models when applied to the fire dataset from our previous search? Two experiments are performed in this paper. The first one was that we ran the state-of-the-art STDM-DL models using their data and compared their performances. Models under this study are trained by either METR-LA or PEMS-BAY dataset, predicting the traffic in spatial and temporal domains. In the second experiment, we used the fire-call dataset of the New Taipei City (NTPC-Fire 2015-17) and implemented some familiar yet straightforward models, such as Autoencoders and GANs, to reconstruct (predict) rasterised heatmaps. Then, we processed the temporal representations using an LSTM-RNN, FBProphet, and ARIMA, to compare performance in time series forecasting of daily and weekly incident frequency. Our first experiment discovered that some state-of-the-art models, such as ST-METANET, STGCN, and Spacetimeformer, all have similar performance. The multi-LSTM outperformed all those models. The "Deepforecast Multi-LSTM" is the best traffic prediction model to date. Surprisingly, in our second experiment, for our small dataset, the FBProphet model outperformed even the best LSTM. Likewise, our best spatial model to reconstruct (predict) a raster heatmap was the Variational Autoencoder (VAE). Given these findings, we further use data visualisation and implement combined models and architectures for each domain in the STDM task. This study suggested that those existing models can be used to solve issues in the spatial-temporal domain, and the use of deep learning networks is a fast-growing research field that depends intensely on big data.

Categories and Subject Descriptors: I.2.8 [ARTIFICIAL INTELLIGENCE]: Problem Solving, Control Methods, and Search – heuristic methods, scheduling.

General Terms: Data Mining

Keywords: Problem Solving, Data Mining, Deep learning, GIS, Geo-Information System, Spatiotemporal Data mining

SECTION 1: INTRODUCTION, BACKGROUND & PURPOSE

This paper is to how deep learning architectures and models can be implemented in ST-data mining in traffic flow prediction, intelligent transport system (ITS) and weather/disaster forecast. And to discover

what group of models, to date, is the best for STDM, especially in traffic flow prediction – such as ST-LSTM and LT-LSTM.

As defined in [17], spatiotemporal data is a dataset related to both space and time. It is collected in the space domain (a.k.a. a (heat) map over a location in the form of raster and vector data matrices) and the time domain (a.k.a. the time-series - a heatmap average for each timestep in the form of timestep-D vectors). Overall, when concatenated, a spatiotemporal dataset is usually a 3D+ tensor.

Given the popularity of GPS-supporting devices, including smartphones, the need for S.T.-data mining rises with the emerging use of intelligent transport systems (ITS, i.e., automatic guidance, self-driving cars, traffic prediction, etc.), and disaster prediction (i.e., weather, flood, earthquake, storms, clouds, smog, global warming etc.).

To manually collect, process, and forecast the ST data is a laborious task. Several models based on both machine learning have been deployed, but ML models need a human to extract the feature representations. Deep learning models, capable of self-feature extraction, usually outperform regular ML ones. Fundamentally, an ST-Deep Learning model works as follows:

- A CNN learns the spatial input (i.e., heatmap, graph data (GCN), & data coordinates)
- While a sequential model, like LSTM/RNN, learns the temporal domain of the input data.
- With both DNNs working together, sometimes with an intermediate submodel (i.e., a dense submodel in [14]), we can correlate and forecast spatiotemporal data (such as traffic, train ridership, etc.) not only the time-series (time-domain) but also the heatmap (space domain).

However, DL models have drawbacks; a CNN/GCN/FCN can only extract and learn only the space domain, while an RNN/LSTM/seq2seq can only do these on the temporal domain of the ST-data. Proven by several papers, RNN, when processing a long sequence, has a "gradient vanishing and explosion" problem. Thus, LSTM-based architectures are preferred in later works.

As a result, some works in section 2 overcame the problems by adding such two neural networks into the same framework, allowing them to extract and learn the ST-data of both domains simultaneously. More technical details are further readable in [8] and [19].

The first part of this paper studies deep learning algorithms and models in ST-data mining & forecasting. The following is a shortlist of spatiotemporal data mining models:

- CNN-GCN-FCNs are used to extract features in the spatial domain (i.e., heatmap).
- LSTM-RNN-seq2seq are used to extract features in the temporal domain (i.e., sequence type data).
- In land usage survey, some researchers use CNN to study changes in land use, crop growth, and construction progresses.
- In ITS, ST-Data mining is extremely useful in traffic prediction, guidance systems, route planning, and self-driving cars.
- In Meteorology, the STDM is used for weather and disaster forecast.
- To make GPS based pathfinding more accurate, Spatiotemporal Data Mining is often utilized in several papers to train the pathfinding ML/DL-architectures.
- In transportation, deep learning methods learn highly intricate ST-correlations among the traffic data – useful in some tasks such as traffic flow prediction, traffic incident detection, and traffic congestion prediction – such as in [13] and [14].
- In On-demand services, such as Uber [10], to perform pathfinding

- All the referred papers involve GIS, Computer Vision (CV), time-series prediction, and spatiotemporal data mining.

According to [8], [17], [19], and [76], there are some fundamentals issues of STDM that needed to be addressed before using deep learning networks:

- Spatiotemporal Data Structures
- Data Instances
- Data Representations
- Deep Learning Models in STDM

SECTION 2: LITERATURE REVIEW & RELATED WORKS

2.1 Notable referred works for early STDM models

Doshi, Basu and Pang [1] developed a CNN model identifying disaster-impacted areas by comparing the change in artificial features extracted from satellite imagery. Using a pre-trained semantic segmentation model from [2] they extracted artificial features, the pre-and post-event images on the "before, during and after" imagery of the event-affected area.

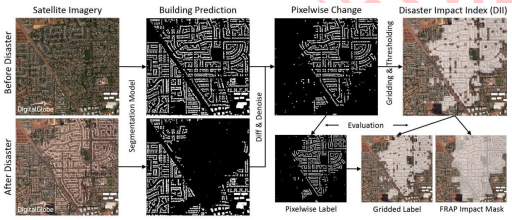


Figure 1: Doshi's Residual Inception Skipnet model for disaster insight [1]

Amit and Aoki [3] proposed a CNN consisting of a sequence of layers, the convolution layer (who detects features from a data image), the pooling layer (downsamples the input), and the FC layer (who classifies the features detected earlier), with ReLU as the primary activation function of the network. Further explained in section 2 of [3].

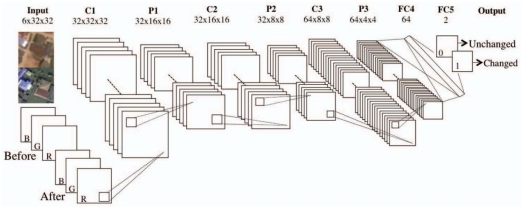


Figure 2: Amit and Aoki's CNN based disaster detection model used in [3]

Iglovikov, Mushinskiy and Osin [4] used an FC-CNN named U-NET and an embedded multispectral sensor, which detects frequency reflection by the objects to detect geo-features in satellite images and yielded satisfying results.

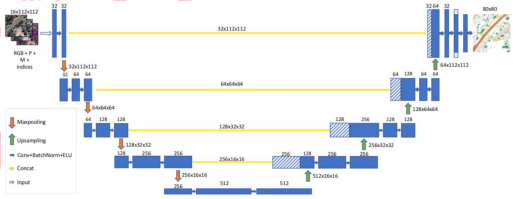


Figure 3: Iglovikov's UNet architecture for geo-feature detection, featuring the downsampling and the upsampling sections [4]

Bochkovskiy, Wang and Liao [5] incorporated YOLO V.4 and TensorFlow Keras into a CNN to improve performance in image recognition. Possibly, we can deploy such a CNN model in this thesis.

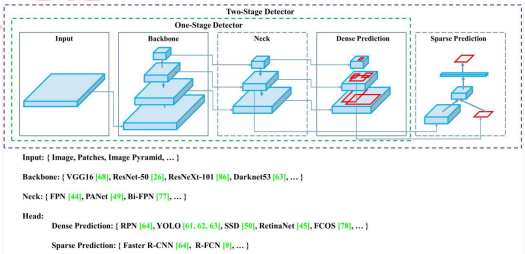


Figure 4: Bochkovskiy's Yolov4 architectural diagram used in [5]

In terms of video and sequence type photos (such as slideshow), however, the use of LSTM-RNN is needed. According to Fang et al. [6], LSTM is excellent at predicting floods because it could process time-series data. In

turn, they designed the long spatial sequential LSTM (LSS-LSTM).

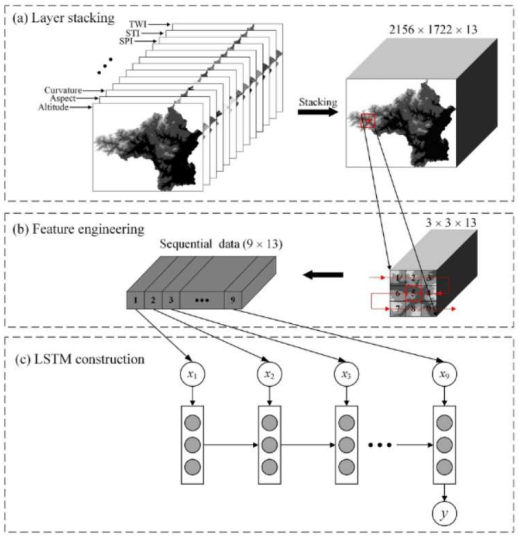


Figure 5: Mechanics of Fang's LSTM model used in [6] to predict flood in Shangyou county, Jiangxi, China

2.2 Notable referred works for graph-based state-of-the-art models

Li et al. [7] view spatiotemporal forecasting as a crucial task for a learning system that operates in a dynamic environment. It can be helpful in pathfinding, autonomous vehicles, logistics, city planning etc. They used a Diffusion Convolutional Recurrent Neural Network (DCRNN) model to forecast the road traffic within a specific space and timeframe. Diffusion convolution extracts the traffic features, and the RNN processes the traffic volumes in sequence. [7]

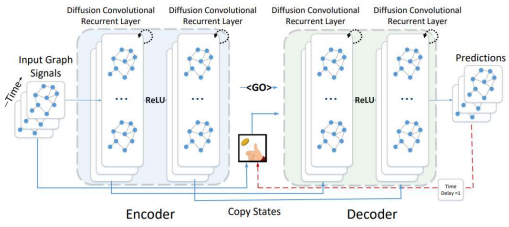


Figure 6: Mechanics of Li's DRCNN model used in [7] to predict the traffic density of each timestamp and location

Wang et al., 2019 [8] surveyed and collected several papers about spatiotemporal data mining. Moreover, they explained the fundamentals and the concepts of STDM. According to the paper and Figure 5, most of the Deep learning STDM models are used for prediction, especially traffic prediction, which comprises most of the works referred to by this paper.

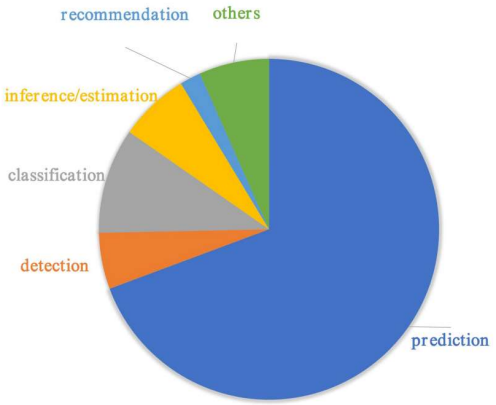
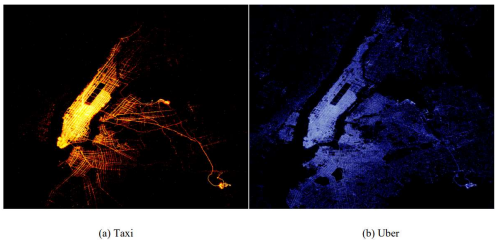


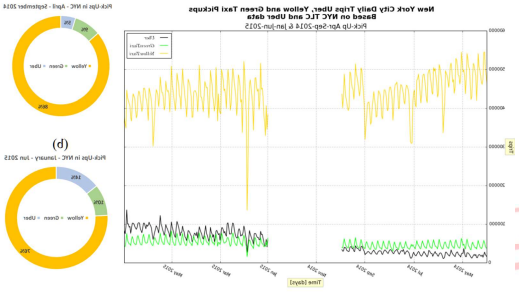
Figure 7: Distribution of the STDM problems addressed by deep learning

Yu et al., 2018. [9], proposed a Spatiotemporal Graph Convolutional Network (STGCN) to tackle the time series prediction problem in the traffic domain. Graph convolution enabled faster training with fewer parameters. STGCN effectively captured comprehensive spatiotemporal correlations through modelling multi-scale traffic networks and consistently outperformed state-of-the-art baselines. [9]

Correa et al., 2017 [10], performed a spatiotemporal data mining of Taxi vs Uber ridership in NYC, 2014+15. According to Figure 8(a), the ridership for both taxi systems depended on several personal factors. With three spatial models for ridership prediction; linear, spatial error and spatial lag models, the last one outperformed the first two algorithms and yielded considerable accuracy and performance. [10]



(a)



(b)

Figure 8: Correa's Taxi-Uber dataset visualization. The spatial module is displayed as heatmap (19a) which is used to predict hailing densities. Meanwhile, the temporal module is displayed as line graph (19b), which is used to predict future ridership volume

Amato et al. [11] designed a deep learning-based architecture called EOF-PCA. The EOF framework decomposes the spatiotemporal input data, obtaining spatial covariates whom the FCNN processes to obtain predictive coefficients, which to be mixed with the decomposed data stream for decomposition, obtaining a spatiotemporal signal reconstruction. [11].

In the ITS, an accurate forecast can forewarn travel outbursts, helping the passengers with their travel plans. Tang et al. [12] designed an LSTM based framework to learn and forecast rail traffic. Architecturally, LSTMs cannot process spatial features, -

- causing them to propose the ST-LSTM architecture. Compared with other conventional models, it performed better in experiments. [12].

Lu et al. [13] designed a spatial-temporal deep learning network, termed ST-TrafficNet. It was evaluated on two benchmark datasets and compared with various baseline methods for traffic flow forecasting. [13]

Pan et al. [14] stated that traffic prediction enhanced traffic safety and the ITS. However, it must face two challenges: 1) complex spatiotemporal correlations of urban traffic and 2) diversity of such spatiotemporal correlations. Owing to these, they designed a deep learning framework for traffic flow prediction called "ST-Metanet" to collectively predict urban traffic in all locations. ST-MetaNet employs a seq2seq autoencoder architecture, comprising RNN for encoding, Meta-GAT for spatial processing, and Meta-RNN for temporal processing. Extensive experiments were conducted based on two traffic datasets to illustrate the effectiveness of ST-MetaNet against several comparable methods.[14] Since the ST-Metanet model is our validating basis, we will put referred results directly in our comparison experiment.

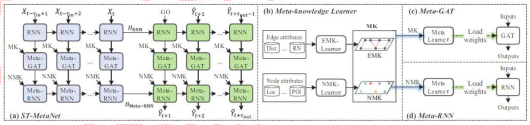


Figure 9: Pan's ST-MetaNet architecture for traffic flow & speed prediction [12]

De Medrano et al. [15] designed A spatiotemporal Spot-Forecasting Framework for Urban Traffic Prediction, named CRANN (Convo-Recurrent Attentional Neural Network). It is highly Adaptable in several ST conditions, easy to understand and interpret, and better & more Stable than state-of-the-art alternatives. [15]

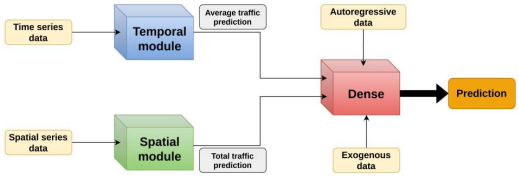


Figure 10: De Medrano's schematics for the CRANN architecture (a) [15]

To cope with the nonlinearity of the traffic flow data during the holidays, Luo et al., 2019 [16] designed a discrete Fourier transform (DFT) and support vector regression (SVR) based machine learning model to predict the road traffic flow during the holidays in Jiangsu Province, China. Properly trained, it outperformed other ML models like ARIMA, SVR, and EMD-SVR. The model is described in the paper itself [16].

Shih et al. 2018 [18] designed an LSTM capable of processing multiple time series at the same time called "Temporal Pattern Attention LTSM" (TPA-LSTM). This RNN-based architecture is designed to process complex and non-linear interdependencies between time steps of multivariate time series data. Attended, this network could remember longer sequences by converting them to the frequency domain for further processing. Regardless of the cases, the model achieved comparable performance with other state-of-the-art models. [18]

Atluri, Karpatne, and Kumar, 2017 [19], like Wang et al. in [8], restated and emphasized the concepts of spatiotemporal data mining.

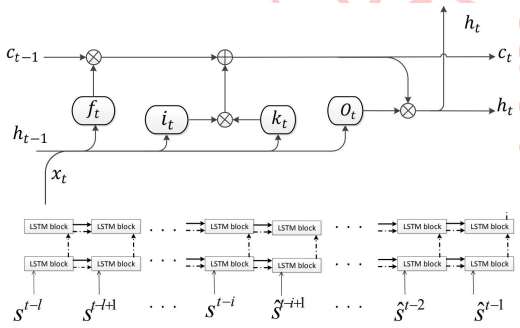


Figure 11: Ghaderi's DeepForecast Multi-LSTM model [73]

Bai et al. (2020) [71] designed a traffic forecasting model named "Adaptive Graph Convolutional Recurrent Network" (AGCRN), which could automatically capture fine-grained spatial and temporal correlations in traffic series. Furthermore, the model outperformed several state-of-the-art models significantly without pre-defined graphs about spatial connections. [71]

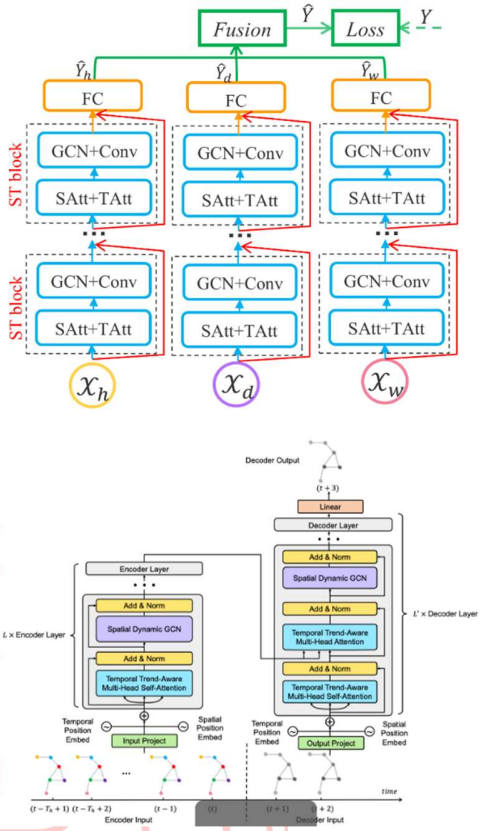


Figure 12: Guo's ASTGCN [72], top and ASTGNN [83], bottom

Guo et al. (2019) designed a traffic forecasting GCN based architecture named "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting" (ASTGCN) [72] and the earlier Attention-based Spatial-Temporal Graph Neural Network (ASTGNN) [83]. Both models comprise of 3-Independent components; recent, daily, and weekly dependencies; and 2-submodels: 1—ST-Attention Mechanism 2. ST-Convolution. It outperformed most contemporary state-of-the-art models. [72].

Ghaderi et al. (2017) [73] designed a framework named "Deepforecast Multi-LSTM". That paper modelled the spatiotemporal data by a graph whose nodes are data-generating entities, and its edges model how these nodes interact. The model can simultaneously forecast all nodes in the graph based on one framework. Compared to other benchmark models, the DL-STF was excellent at short term forecasts. [73]

For attentive models, Grigsby, Wang, and Qi (2021) [80] designed a transformer-based STDM architecture by connecting an attentive seq-seq model (for the temporal domain) to an attentive GCNN (for the spatial domain) for multivariate ST-data forecasting on the NY-TX weather and the METR-LA traffic datasets and achieved outstanding results. Unfortunately, in our comparative experiment, the model did not make it to the testing phase after being trained for 80 epochs (8 hours in a 4x GPU TWCC container) due to a GPU memory burst, causing it to be another basis. [80]

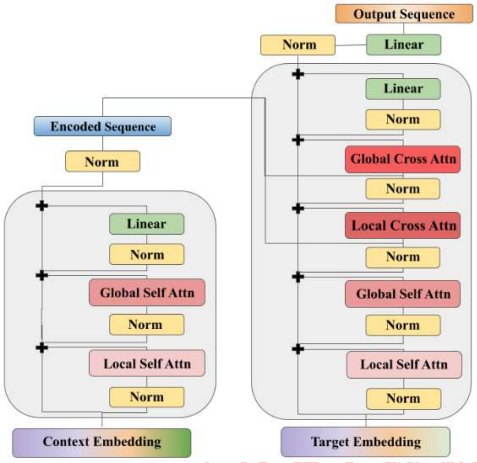


Figure 26: Grigsby's Spatiotemporal Transformer model [80]

2.3 Notable referred works of models suiting non-graph event data

To apply the model design process to non-graphical event data, such as dot-and-hotspot event datasets, we must first split the dataset into the spatial domain (rasterised heatmaps) and the temporal domain (time series of frequency by timeframe). Most of the models are written in Python with the Tensorflow-Keras library.

For the spatial domain, the most fundamental models used in our paper as a reference for the study are from the TensorFlow-Keras website [74] and Borwnlee's blog [84], including Autoencoders [74] and GANs [75]. Most of our generative architectures are from this source: [78], notably DCGAN [79], LSGAN [82] and Wasserstein GAN (WGAN) [74]. Most of the spatial models referred are designed based on the MNIST (28*28) dataset, prompting to rasterise the spatial domain to an album of 28*28 images.

For the temporal domain, the LSTM-RNN model is the most popular. Moreover, Facebook Prophet (FBProphet) [76] is also a powerful model for small datasets, rivalling the LSTM in terms of performance metrics. For Example, Menculini et al. [78] designed a wholesale food price prediction model by comparing ARIMA, FBProphet, LSTM, and CNN-LSTM in their performance. Due to the data's apparent seasonality, they found out that FBProphet did not perform well as expected, while ARIMA rivalled a simple LSTM – with CNN-LSTM model outperforming them all due to feature pre-extraction functionality of the CNN.

2.4: Applications of Deep Learning Models in STDM

Restating what we have said in Section 1, the paper [8] and [19] covered several fields of applications, such as climate science, neuroscience, social sciences, epidemiology, transportation, criminology, and Earth sciences. Here are some GIS application examples:

- **Meteorology:** Prediction of poor weather and disasters assist in reducing life and property loss. For example, the use of CNN is found in [1], [2], and [3]; as well as LSTM found in [6], as well as Zhang et al.'s, 2018 [48], designed a Multilayer Perceptron Neural Network (MLPNN) for short term rainfall forecasting in China and achieved considerable performance – outperforming ARIMA and SVM.
- **Transportation:** Corresponding to Figure 7, STDM has the most popular application in transportation. With the rising availability of transportation data collected from various sensors, there is an urgent need to utilize deep learning methods to learn the complex and highly non-linear

spatiotemporal correlations among the traffic data to facilitate various tasks. Traffic flow, direction & speed prediction is the most popular application of this field, e.g. [7], [9], [12], [13], [14], [15], and [16].

- **On-Demand Service:** Taxi-hailing apps are becoming popular. One of the notable works for taxi usage prediction is [10].
- **Human Mobility:** Mining the human mobility data is practically important for applications, including traffic forecasting, urban planning, and human behaviour analysis. Many recent works use deep learning methods for urban crowd flow prediction and crowd density estimation; for example, Zhang et al. 2018 [49] designed an ST-ResNET based model to predict citywide crowd flow in Guiyang, Beijing, and New York City.
- **Location-Based Social Networks (LBSNs):** Some social network platforms, such as Foursquare and Flickr, use GPS features to locate the users and let the users broadcast their locations and other contents from their mobile device. Various deep learning models are currently designed to analyze the user-generated ST data in LBSN; for example, Zhao et al. 2018 [50] designed an ST-LSTM model to recommend POIs for Foursquare and Facebook users.
- **Criminology:** A criminal ST data consists of location coordinates and timestamps. DL models are set to predict crime incidence on a heatmap; for example, Duan et al. [51] proposed a Spatiotemporal Crime Network based on CNN to forecast the crime risk of each region in the urban area for the next day.

2.5: Summary of section 2

Furthermore, we also categorized referred papers into several application categories and discovered that most of them belong to the traffic prediction category. To sum up, most models referred to, designed for traffic prediction work, are built with complex DL architectures, which improved accuracy and reliability over basic models. DL-based architectures also contribute to the data mining of different learning approaches, which are mentioned in section 3.1. Nevertheless, to apply the DL models on our non-graph dataset, we need to implement models referred to in section 2.2 in our second experiment, the custom event heatmap experiment (CEHE).

SECTION 3: METHODOLOGY

In this section, first, we will list different approaches to STDM. Then, we will discuss the advantages and disadvantages of those approaches. Next, we will experiment with applications of the STDM process to the NTPC-fire dataset, a dot-and-hotspot event dataset. After that, different performance measures for STDM will be compared, and we will try to address the effects of those measures. Lastly, we will conclude this section with characteristics for different STDM processes.

3.1 Different approaches to spatial-temporal data mining

In some studies, such as [8] and [19], ST-Datasets were mined using both traditional Machine learning methods – such as Decision tree, DBSCAN, SVM, ARIMA, etc. And deep learning methods – CNN, RNN, LSTM, Transformers, etc. As listed below, we summarize different approaches to ST-datasets.

- **Clustering:** Clustering refers to grouping instances in a dataset that shares similar feature values. **In STDM, clustering can be performed on:** Points, trajectories, time-series, spatial maps, and ST raster data**. Nevertheless, a challenge must be overcome;** clustering locations based on their time series must ensure that the discovered clusters are spatially contiguous. Ignoring this can lead to location data misinterpretation. There are **some notable clustering algorithms, namely** ST-DBSCAN [20], clustering ST-points using DBSCAN algorithm [21], CLARANS [22], 'dynamic ST clusters' [23]. Etc.

- **Prediction:** Predictive learning learn a mapping from the input features to the output variables using a representative training set. In spatiotemporal applications, both the input and output variables can belong to different types of ST data instances.
  - **Time-series:** RNN and LSTM are widely used for time-series data prediction. The weather variables such as wind speed are usually modelled as time-series, and then RNN/LSTM models are applied for future weather forecasting [24], [25], [26], [27], [28], [29].
  - **Spatial maps:** The spatial maps can be usually represented as image-like matrices and thus are suitable to be handled by CNN for various predictive learning tasks [30], [31], [32], [33]. One of the examples involving spatial map prediction is of Zhang et al., 2016 [32]. They proposed a CNN based crowd flow forecasting model called UrbanFlow for real-time urban crow flow prediction. In order to capture the temporal and spatial correlations of a sequence of spatial maps simultaneously, many works tried to combine CNN with RNN for the prediction. This method is utilized in [6], [7], [14], [15], and [18]. Generative models dedicated to rasterized heatmaps [74], such as Autoencoders and Generative Adversarial Networks (GANs), are preferred in our work to predict a heatmap of a disaster, such as building fires.
  - **Trajectories:** Currently, two types of deep learning models, RNN and CNN, are used for trajectory prediction depending on trajectorial data representations. Trajectories are a sequence (learnable by RNNs) type data of locations representable as a matrix (learnable by CNNs).]. One of the examples involving trajectorial prediction is of Lv et al., 2018 [34]. They modelled trajectories as two-dimensional images, where each pixel represented trajectorial locations. Then an MLCNN were adopted to combine multi-scale trajectory patterns for destination prediction of taxi trajectories. Furthermore, some works also incorporate GPS, such as [35], who proposed a 4-layer LSTM model named DeepTransport, learning a set of GPS trajectories, to predict different transportation modes
  - **ST Raster:** ST raster can be represented either as matrices dimensioned as location and time or 3D tensors dimensioned as cell region ID, cell region ID, and time. [8] Usually, 2D-CNN (matrices) and 3D-CNN (tensors) are applied for ST Raster data prediction, and sometimes they are also combined with RNN. 3D-CNN models are purposed in [36], [37], [28] and [38]. For example, Rasp & Lerch, 2018 [28], modelled the mobility events of passengers in a city in different time slots as a 3D tensor, then they used a 3D-CNN model to predict the supply and demand of the passengers for transportation. The major difference between an ST Raster and a spatial map is that the ST raster is the merged measurements in multiple time slots, while the spatial map is the measurement in only one timeslot. While preparing our NTPC-fire dataset, we rasterised our spatial map with dot data by loading event dots into each raster cell to obtain their concentration using our program's "cytogenesis" function.
  - **Using Temporal Information (Classic ML) [19]:** There are some estimations of time-series' future trends, such as exponential smoothing techniques (ESTs) [39], ARIMA models [40], and state-space models [41].
  - **Using Spatial Information (Classic ML) [19]:** We surveyed a vast body of literature on spatial prediction methods that consider the spatial auto-correlation structure in the data to ensure spatially coherent results, such as spatial autoregressive (SAR) models [42], geographically weighted regression (GWR) models [43], and Kriging [44] Markov random field-based approaches [45], [46], [47].
- ***Representation Learning:*** *RL-models* aims to learn the abstract and useful representations, formed by (non)linear

transformations compositions of input data, of the input data to facilitate downstream data mining or machine learning tasks

- **Trajectories**: Trajectories are ubiquitous in location-based social networks (LBSNs). CNN, GCN, GRU and RNN are used to learn such representations. For example, Ding et al, 2018 [52], proposed a geographical convolutional neural tensor network named GeoCNTN to learn the embeddings of the locations in LBSNs.
- **Spatial maps:** There are several works studying how to learn representations of spatial maps. CNN, GCN, and Autoencoders are used in this learning method. For example, Costilla-Reyes et al., 2017 [53] proposed a CNN architecture for learning ST features from raw spatial maps of the sensor data.

- **Classification:** The classification task is mostly studied in analyzing fMRI data for disease identification. In Neuroscience, for instance, [55] employed an LSTM to diagnose autism spectrum disorders (ASD) using fMRI time-series data. Furthermore, [56], [57], [58], [59], [60], [61] modelled the fMRI data as spatial maps and then used them as the input of the classification models. Feature classification & recognition is also present in [3], [4], and [5], which used a CNN based architecture to classify features in satellite images.
- **Estimation and Inference:** Current ST-data estimation and inference works mainly focus on spatial map and trajectory data types.
  - **For spatial maps,** while monitoring stations have been established to collect pollutant statistics, the number of stations is usually minimal due to the high cost. One of the models utilizing this method is the ADAIN architecture, which [62] applied for modelling air quality inference of any location-based on pollutants and learning some complex feature interactions.
  - **For trajectories,** STDM of this purpose is used along with GPS, which is notable in [10]. Another example is of Zhang et al., 2018 [63]. They proposed an RNN-based deep model named DEEPTRAVEL, which can learn from the historical trajectories to estimate the travel time.
- **Anomaly Detection:** Anomaly detection or outlier detection aims to identify the rare items, events or observations that differ remarkably from the rest of the data. For instance, ST-DBSCAN is used in ST point anomalies [64], [65] and [66]. Moreover, Anomaly detection is also used in disaster analysis & prediction in [1] and [6]. DBN and LSTM are used for event data on traffic accident tweets [67]. For example, Zhu et al., 2018 [68] proposed utilising Convolutional Neural Networks to automatically detect traffic incidents in urban transportation networks by using traffic flow data. By the way, **for spatial maps,** ST-CNN is implemented in [69] for extreme weather events identification.
- **Other notable methods:** there are some other applications, such as change detection [19], pattern mining [8], relation mining [8], [19], and POI recommender systems, [8] whose example is that Wang et al., 2019 [70] proposed an attention-based-RNN for personalized route recommendation.

3.2 Advantages and disadvantages of applying different approaches

Here, we will explain why deep learning models are preferred in recent works over traditional machine learning models, which can be seen more in Table 1:

Table 1: comparison of DL vs ML methods in STDM. Source: [8]

	Deep Learning Models	Traditional ML models
ST Features	Automatic Learning	Hand-crafted
ST Data types	n-to-n	1-to-1
STDM Tasks	Prediction, classification, estimation, etc.	Prediction, classification, estimation, clustering, frequent pattern mining, change detection, etc.
Temporal dependency	Long/Short term	Short term
Spatial dependency	Global/Local	Local
Interpretability	Low	High
Domain Knowledge	Little	Much

According to Table 1, we conclude that DL models are chosen over ML ones due to...

- **Automatic feature representation learning:** Unlike ML models, which can only handle hand-crafted ST-features, DL models can automatically learn hierarchical feature representations from the raw ST-data. Multilayer convolution in CNN and recurrent structure in RNN can effectively learn highly complex ST-data.
- **Powerful function approximation ability:** Each DLNN has multiple layers; consisting of nonlinear modules with convolution, pooling, dropouts, and activation functions. With the composition of enough such transformations, very complex functions can be approximated to perform difficult STDM tasks with complex ST data.
- **Performing better with big data:** Traditional ML methods, e.g., SVM, Decision Tree, perform better on small datasets but become ineffective when the training dataset enlarges. However, DL models improve their performances when they learn larger datasets due to their powerful feature learning and function approximation abilities.

3.3 Different measures of performances

Usually, each DL framework has its performance measured in mean squared error (MSE), mean absolute error MAE), and root mean squared error (RMSE). Some models, however, used different performance metrics, such as that of Shih et al. 2018 [18], which had some unfamiliar metrics, like RAE, RSE, CORR, etc. Thus, his code needs an overhaul to ensure that both the input and the output tensor dimensions are equal; to calculate MSE. The experiment, divided into two episodes, has been further described in section 4.

In our first episode of the experiment, from each referred/surveyed paper, primarily on Google-COLAB environment with TWCC as the backup environment, which, if a model exceeds the COLAB's resource limits, we will run its Python code there. Then, using their original datasets, we will run the models we have referred to in section 2.2 and compare them in the form of MSE, RMSE, and MAE metrics; the less of them for each model, the better it performs in the term of learning and forecasting. Finally, we will conclude which model does the best learning/forecast. As this is a survey thesis, we will not design a custom model.

In our second episode, given an event map NTPC-fire dataset, we will visualize and split the dataset into the spatial (by converting it into a raster heatmap) and the temporal domains (by converting it into a time series), and choose some familiar yet straightforward models to learn and forecast the data, and do the analytics same as the first episode.

3.4. Hypothesis:

To hypothesise our research, we have asked ourselves two primary questions:

3.4.1. STDM DL-models can do a variety of learning and predictions, but how well can they do?

- Metrics: MSE, RMSE, & MAE
- What architecture is the best model in STDM?
- Is the ST-METANET model still the best?
- Do newer, more architecturally advanced models always outperform the older ones?

3.4.2. If we have a custom dataset with its data structure visualised, which model to be learned from it?

- Such as the "NTPC-fire_2015-17" dataset which is a small event dataset
- Do we need to design a new model?
- If we implement existing models to train such a dataset, which model makes the best prediction for each domain?
- Of the same model, what will happen if we vary some hyperparameters?

3.5. Contributions

To be seen in section 4, we have studied several STDM papers and proved their model interesting and applicable. Furthermore, we have run five models: Keras-LSTM, CRANN, DCRNN, Multi-

LSTM and ST-METANET; they all performed well for their architectures. More contributions can be read in Section 4.4.

Also, we ran generative models, like AEs and GANs, to predict our rasterized map and predicted the trends using time-series forecasting models (like LSTM-RNN, FB-prophet, and ARIMA), along with performing some subtests which will eventually lead to hyperparameter tuning.

SECTION 4: EXPERIMENTS

4.1. Experiment Settings for Referred Papers & Datasets (a.k.a. Exploratory Comparative Experiment; ECE)

4.1.1 ECE settings

Our experiment[3] is divided into two episodes; one is for referred models for comparison called "exploratory comparative experiment", abbreviated ECE, and another one is of our "New-Taipei-City-fire" dataset for heatmap generation and incident prediction called "custom event heatmap experiment", abbreviated CEHE. The list below displays potentially useful STDM models used in the first episode.

Continuing from section 3.3, the deep learning models and ANNs are written in Python with Pytorch, Mxnet, and Tensorflow-Keras as our preferred DL libraries. Be advised; not every model is runnable on the stock COLAB environment and thus must be modified to meet their requirements. If a model consumes more computing resources than what COLAB can provide, it is advised to connect COLAB with a cloud VM instance or a physical Linux machine with RAM around 64 GB, for example, an EC2 Linux instance with GPU. In our case, the viable backup environment is TWCC. All lost functions are tied to their architecture, with formulae explicitly written in their referred papers.

For the Taxi-Uber-Simple LSTM models, which process only the temporal domain, the Keras simple LSTM is in the I-H-O formation of 1-4-1 (same as one in our custom dataset experiment), whilst the Pytorch simple LSTM has the formation of 1-2-2.

Only in this episode, for each model, the dataset used to train it is the same as their respective original papers. Their hyperparameters would be the same as their default settings unless changed, with Adam as their optimiser. The primary metrics used to measure the models' performance are MAE, MSE and RMSE. The models' settings are defined in Table 3. The blue characters denote models lacking a suitable environment, whose results must be directly referred from their papers.

Table 2: Models and datasets in the comparative experiment – brief table

Architecture	Dataset
LSTM-Pytorch	Taxi-Uber Dataset (NYC, 2014-15)
LSTM-TF/Keras	Taxi-Uber Dataset (NYC, 2014-15)
CRANN	Madrid traffic & weather dataset (2018-19)
ST-Metanet (Flow)	Taxi-Flow (Beijing, 2010)
ST-Metanet (Speed)	METR-LA (Los Angeles, 2014)
DCRNN	METR-LA & PEMS-BAY
AGCRN	Caltrans PEMS04&08
ASTGCN	Caltrans PEMS04&08
STGCN	METR-LA & PEMS-BAY
Spacetimeformer	METR-LA

[3] Available in github.com/Suppersine/Thesis2021/tree/main/notebooks

4.1.2. ECE Results.

In Table 4, the green cell indicates the best model to date. The yellow highlights indicate partially successful models. The blue characters indicate possible metric values as if whose models were operational (a.k.a. basis), referenced by the numbers, which can be turned black later if a suitable environment is found. The black characters indicate models run with COLAB. The red characters indicate models run in TWCC containers. The pink cells indicate models with memory problems – insufficient RAM of either the mainboard or GPUs – requiring a suitable (64+GB RAM) environment to run. Despite stringent environmental restrictions preventing such colour-marked models from running on COLAB, it was evident that the models performed well, at least for their architectures. To see the detailed issues and viable solutions, see section 4.4.

The AGCRN, the STGCN, and the ASTGCN models suffered a serious dataset error, even if running on COLAB Nevertheless, we managed to run the model successfully with intensive "debugging and troubleshooting" noted in our "ipynb" notebooks. With a significant overhaul on COLAB, we can finally run but did not completely retrain the DGCRN model, luckily due to the saved/pre-trained weights in its GitHub repository. For the ST-METANET model, 64GB+ RAM is a safe level - as we cannot afford to rent a suitable cloud VM for it, we decided to put the model as a basis, directly referring to the original results in pink cells – the same measure also applies to Spacetimeformer owing to the same error occurring to the GPU.

In Table 4, the simple LSTM models had a tremendous RMSE error in thousands due to the model's simplicity, overfitting, and gradient explosion. The UBER-LSTM Pytorch model is somewhat accurate because the Pytorch-LSTM model is perfectly hyperparameter-tuned for time-series with apparent seasonality. Nevertheless, we should never trust the numbers unless we see the graphs of Figures 17 and 18. The CRANN Model, despite setting to 200-epoch maximum, automatically finished training around 100 epochs. Using TWCC instance with 2x GPUs, after 2 hours of training, the AGCRN produced numerically the same results as its paper - however, for the same model run on a COLAB VM, if able to, the results are more divergent from the reference due to its inferior nature compared to paid VMs.

It is also evident that when learning correct datasets, complex state-of-the-art models (such as CRANN) vastly outperformed simple models (like Keras-LSTM). As we can see in Table 4, the "Deepforecast" Multi-LSTM model, to date, is the best model for spatiotemporal traffic prediction due to the depth of a classic but straightforward to calibrate LSTM – nothing could match it when trained by MS-Winds dataset. Nevertheless, it takes 2 hours to train in COLAB.

A rivalling model to Deepforecast is DCRNN trained by PEMS-Bay dataset, with both static (STA) and variable (VAR) modes, which has a comparable performance; like 1.16 vs 1.6(STA) or 1.74(VAR) MAE, and 1.58 vs 3.43(STA) 3.09(VAR) RMSE.

4.2. Experiment on our Custom Dataset. (a.k.a. Custom Event Heatmap Experiment; CEHE)

4.2.1 CEHE Experiment Settings & Details

In this (2nd) episode, we have an event dataset (with coordinates & timestamps) of fire incident record data of New Taipei City, Taiwan, and we will choose referenced models. To start, we selected all models in section 2.2, trained them with the NTPC fire data, and ran it under the Google-COLAB environment. Each ".ipynb" notebook contains the data input and processing section. We started with the dataset having its dots, recording both timestamps and coordinates, extracted. Then, the "cytogenesis" function, fed with the extracted data and a square map width,

converted dots to raster heatmap, producing 36 images, each representing each month in the 2015-17 timeframe. The testing heatmaps were March, June, September, and December, while the training data belonged to other months. For the temporal domain, however, each timestamp is extracted and has a frequency of 1 day – when each timestamp is grouped in a timeframe, be it daily, weekly, or monthly, the frequency also counts against a timeframe, obtaining a time-series recording incident frequencies by timeframe. In our time-domain models, only the daily series is used for prediction unless specified by other timeframe widths. All models had the Adam optimizer with a learning rate of 0.001 and were trained for 100 epochs. Table 5 shows its preliminary results – the yellow cells display the best results available.

In one of our hyperparameter subtests, we increased the batch size to 12 for all Spatial models, except for the BAE. Consider viewing Table 5 for the results.

4.2.2 CEHE Results

Supposedly, more complex and advanced models usually outperform the simple, basic ones. As shown in Table 5 and Figure 13, for the autoencoders, due to simplicity, and well-tuned hyperparameters despite minimal touches, of VAE over CVAE when trained by our tiny dataset, the VAE outperformed the CVAE. It is ironic that convolution in the VAE, i.e., CVAE, does not always produce better MSE (but still better MAE) than the VAE. Nevertheless, both autoencoders outperformed their basic counterparts (BAE). Usually, GANs often outperform Autoencoders, but it is not always true in our case due to our small dataset size. Some exceptions are DCGAN & WGAN GP out-MAE'd the BAE. Interestingly, LSGAN is much harder to train than WGAN and DCGAN proven by its notably poor performance when trained by our small dataset of rasterised heatmaps as it did not finish the training due to the architecture designed for a low learning rate. In Table 5 of the time domain, try to notice hidden hyperparameter subtests, which is to be further analysed in section 4.4.

For the Temporal domain, it is usual that FBProphet and LSTM outperform ARIMA, even with the same frequency. Notably, for our small non-seasonal time-series, FBProphet outperformed the LSTMs because this architecture did not overfit; and never overfits. Furthermore, we have observed that the PreMMS-LSTM, not yet overfit, can still be trained and hyperparameter-tuned to overtake the FBProphet model. With overfitting occurring on the PostMMS[4]-LSTM, it became so saturated that it could not be trained further. Nevertheless, we carried out hidden hyperparameter subtests with it. Furthermore, adding some lookback steps to the LSTM also helped with training, improving test metrics. Our future suggestion is to prioritise the PreMMS-LSTM and do some hyperparameter subtests with it.

The PreMMS-LSTM normalises the data before the train/test split (TTS), while the PostMMS one does it later after the TTS.

To date, the FBProphet model in the temporal domain and the VAE in the spatial domain are our best models trained with our NTPC-fire dataset. For visualized results, Figure 14 denotes all rasterized heatmaps generated by each spatial model, and Figure 15 displays the temporal prediction results.

According to Table 5, Table 6, and Figure 13, upon adding more images to the training batch, , the DCGAN and the LSGAN improved, and the others worsened. The CVAE and the VAE rivalled the MAE values, with CVAE producing better MAE most of the time. The BAE did not enter this subtest. Nevertheless, the VAE is still the best model for our dataset. Only for PostMMS-LSTMs, graphs were plotted from June 1, 2017, onwards. Visualised predictions of Table 5 can be viewed in Figures 14, 15 and 16

[4] MMS = Min-Max-Scailing; the prefix informs when processed, before or after the TTS.

Table 3: Models and datasets in the comparative experiment – plus settings

Architecture	Architecture Description	Dataset Name/Desc	learning rate	Epochs	Major DL Module
Taxi-Simple-LSTM-Pytorch	Simple-LSTM	Time series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Uber-Simple-LSTM-Pytorch	Simple-LSTM	Time series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Taxi-Simple-LSTM-Keras	Simple-LSTM	Time series of Taxi-Uber DS. (2014-16)	0.001	100	TensorFlow Keras
Uber-Simple-LSTM-Keras	Simple-LSTM	Time series of Taxi-Uber DS. (2014-17)	0.001	100	TensorFlow Keras
CRANN-Temporal	Bahdanau Att.Mech Autoencoder (LSTM based)	temporal time series of hourly/daily car traffic (in Madrid)	0.01	200	Pytorch
CRANN-Spatial	CNN+ST-Att.Mech	graph data captured by 30 sensors + Timestamps (A 17000x30 matrix)	0.01	200	Pytorch
CRANN-Dense	Fully Connected Feedforward NN (FCFFNN)	dense 3D+ tensor of both preceding modules	0.01	200	Pytorch
Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
GAT Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
ST-Metanet (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
GAT Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
ST-Metanet (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
AGCRN	Attentive Graph CRN	Caltrans PEMS04&08	0.003	100	Pytorch
ASTGCN	Attention Based GCN	Caltrans PEMS04&08	0.001	80	Pytorch
Deepforecast	Multi-LSTM	MS_winds - Wind Speed & Flow Dataset	0.001	80	TensorFlow Keras
DCRNN	R-CNN	PEMS & METR-LA	0.001	100	TensorFlow Keras
STGCN	Graph-CNN	PEMS & METR-LA	0.001	50	Pytorch
Spacetimeformer	Transformer opted for ST-data	METR-LA	0.001	80	Pytorch

Table 4: Experiment results of the state-of-the-art models (the ECE episode)

		Key Metrics		Other Metrics	
Architecture	**MSE**	**RMSE**	**MAE**	**Type**	**Value**
Taxi-Simple-LSTM-Pytorch	72269860	8501.168	5753.569		
Uber-Simple-LSTM-Pytorch	0.0332864	0.1824457	0.13018544		
Taxi-Simple-LSTM-Keras	93431556	9666	92.484945		
Uber-Simple-LSTM-Keras	105050200.4	10249.4	83.21		
CRANN-Temporal	6653.6636	81.569992	50.3727684	Bias	-3.82883501
				Relative error %	24.85518008
CRANN-Spatial	55740.719	236.09472	117.7406616	Bias	-14.2317371
				Relative error %	9.053748846
CRANN-Dense	67264.055	259.35315	138.2879333	Bias	-4.71734381
				Relative error %	24.85518008
Seq2seq (flow) [14]	1626.986623	40.33592224	21.3		
GAT Seq2seq (flow) [14]	1098.041371	33.13670731	18.3		
ST-Metanet (flow) [14]	813.1885148	28.51646042	16.9		
Seq2seq (speed) [14]	44.4751941	6.668972492	3.55		
GAT Seq2seq (speed) [14]	36.92343427	6.076465607	3.28		
ST-Metanet (speed) [14]	33.63158961	5.799274921	3.05		
AGCRN - PeMSD4	1040.761367	32.26083333	19.69416667	MAPE (TWCC)	13.02040833
AGCRN - PeMSD8	718.9101563	26.8125	17.01583333	MAPE(COLAB)	10.65138333
ASTGCN - PeMSD4	1173.7476	34.26	21.84	MAPE	0.15
ASTGCN - PeMSD8	809.4025	28.45	18.5	MAPE	0.11
Deepforecast	2.481881285	1.5753988	1.1590073	NRMSE_maxmin(%)	15.575216
				NRMSE_mean(%)	43.097104
DCRNN(Metr-LA)-STA - 15min	75.5161	8.69	4.02	MAPE	9.39
DCRNN(Metr-LA)-STA - 1hr	201.9241	14.21	6.79	MAPE	16.71
DCRNN(Metr-LA)-VAR- 15min	60.5284	7.78	4.37	MAPE	10.08
DCRNN(Metr-LA)-VAR - 1hr	114.0624	10.68	6.5	MAPE	15.84
DCRNN(Pemsbay)-STA - 15min	11.7649	3.43	1.6	MAPE	3.24
DCRNN(Pemsbay)-STA - 1hrmin	49.2804	7.02	3.05	MAPE	6.84
DCRNN(Pemsbay)-VAR - 15min	9.5481	3.09	1.74	MAPE	3.59
DCRNN(Pemsbay)-VAR - 1hr	26.1121	5.11	2.92	MAPE	6.46
STGCN - 15min	16.532356	4.066	2.231	%Wmape	5.206
STGCN - 30 min	33.051001	5.749	3.04	%Wmape	7.386
STGCN - 45min	46.744569	6.837	3.623	%Wmape	8.927
Spacetimeformer [80]	36.21	6.017474553	2.82	MAPE	7.71
				model.loss	0.258

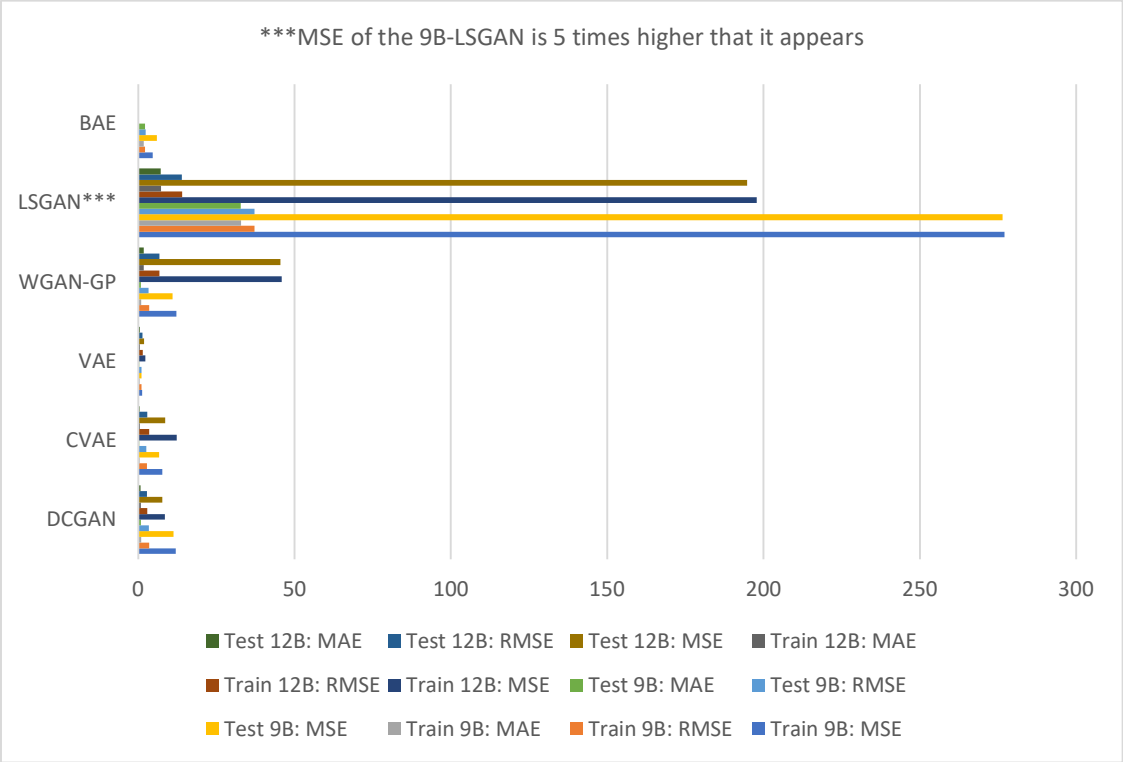


Figure 13: Metrics of CEHE spatial models

Table 5: Models and results in the NTPC fire CEHE (both domains)

MODEL	DOMAIN	Batch size / frequency	Train Metrics:				Test Metrics:			
			MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
Pre-MMS-LSTM4	Time		91.769263	9.579627	5.283182	0.248342	52.195229	7.224627	5.508191	0.37734
Post-MMS-LSTM4-noLB	Time	1day +1day-lookback	7.679134	2.771125	1.556069	0.133908	61.438013	7.83824	5.954068	0.29369
Post-MMS-LSTM-LB5	Time	1day +5day-lookback	7.582465	2.753628	1.559106	0.138684	57.692435	7.595554	5.70559	0.283949
BAE	Space	24 train / 12 test / 9 show	4.636669	2.1532927	1.8429354	1541306600	5.944328	2.4380991	2.1663473	1860365700
DCGAN	Space	12	8.476757	2.91148709	0.80054784	38606776	7.6664495	2.76883541	0.7418378	38433640
Fbprophet (37 days)	Time	1 day	n/a	n/a	n/a	n/a	48.613108	6.97231	5.045342	0.281977
CVAE	Space	9	7.6662908	2.7688065	0.3827354	177096430	6.7336392	2.5949256	0.3442523	174446380
VAE	Space	9	1.3036826	1.14178919	0.37329638	4.11115E+14	1.0857255	1.04198155	0.3522904	4.2458E+14
WGAN-GP	Space	9	12.2007065	3.49295097	0.8706569	8.83334E+12	10.965601	3.31143488	0.81602687	8.84299E+12
Weekly ARIMA	Time	7 days	n/a	n/a	n/a	n/a	1436.90057	37.9064714	28.42712296	0.195063926
LSGAN	Space	12	197.77995	14.0634261	7.245772	2.64E+16	194.74649	13.95516	7.1890564	2.63E+16
Post-MMS-LSTM-Weekly	Time	7 days	227.370864	15.078822	10.250484	0.086445	852.383545	29.195608	21.007481	0.143819
Post-MMS-LSTM9	Time	1day +5day-lookback	7.578619	2.752929	1.563434	0.139068	55.893509	5.60772	5.70559	0.281344

Table 6: Models and results in the CEHE: the spatial domain with batch sizes of 9 to 12

		DCGAN	CVAE	VAE	WGAN-GP	LSGAN***	BAE
Train 9B:	MSE	12.011371	7.6662908	1.3036826	12.2007065	1385.0496	4.636669
	RMSE	3.46574244	2.7688065	1.14178919	3.49295097	37.21625398	2.1532927
	MAE	0.8862971	0.3827354	0.37329638	0.8706569	32.86514	1.8429354
Test 9B:	MSE	11.25666	6.7336392	1.0857255	10.965601	1382.266	5.944328
	RMSE	3.3550947	2.5949256	1.04198155	3.31143488	37.17883795	2.4380991
	MAE	0.8440172	0.3442523	0.3522904	0.81602687	32.839905	2.1663473
Train 12B:	MSE	8.476757	12.374425	2.326788	45.895508	197.77995	4720.15986
	RMSE	2.91148709	3.5177302	1.52538125	6.77462234	14.06342608	95.3192812
	MAE	0.80054784	0.48861146	0.5209149	1.813967	7.245772	0
Test 12B:	MSE	7.6664495	8.672948	1.9275316	45.41533	194.74649	0
	RMSE	2.76883541	2.9449868	1.38835572	6.73908963	13.95515999	0
	MAE	0.7418378	0.41251573	0.48643875	1.7555918	7.1890564	0

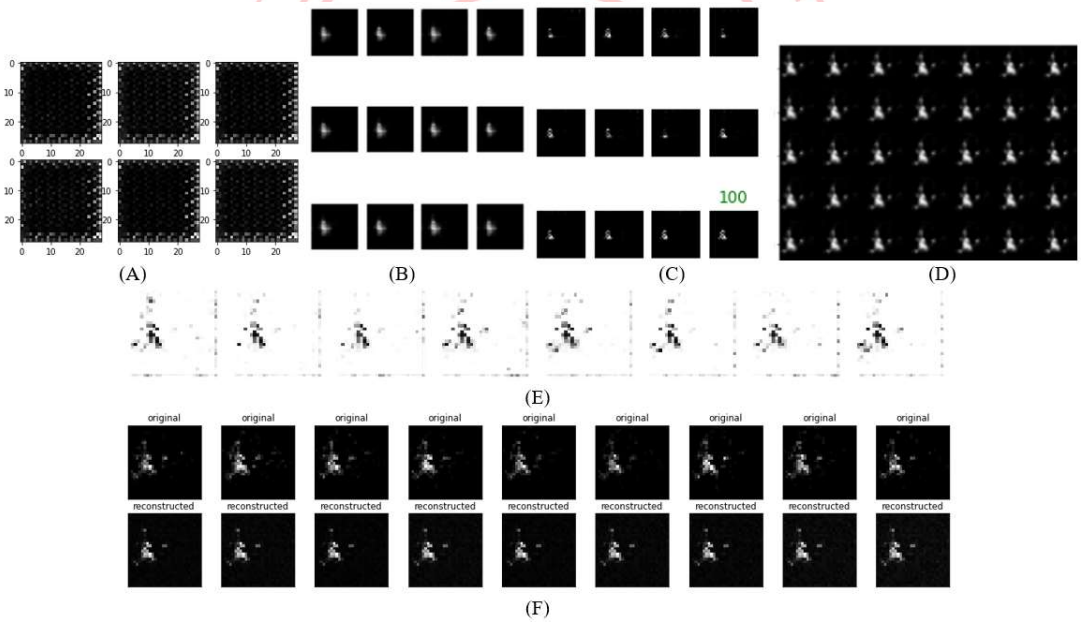


Figure 14: The visualised NTPC-Fire raster heatmap prediction generated by some different models: (A) = LSGAN, (B)=CVAE, and (C)=DCGAN. (D)=VAE, (E)=WGAN-GP, (F)=BAE with the input on the top. After 100 epochs, we can see that VAE and BAE were not only easy to train, but also produced realistic reconstructions.

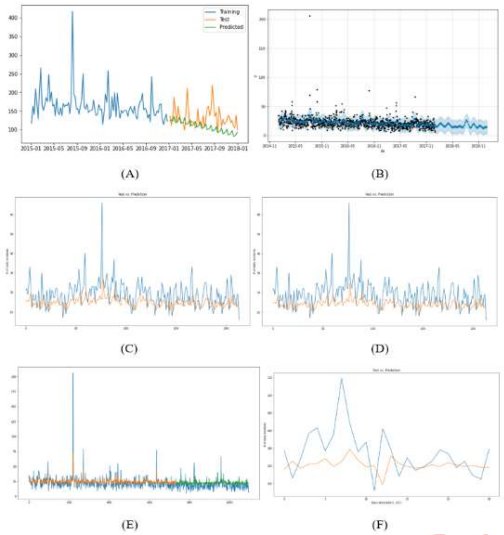


Figure 15: The visualised time-series prediction of some different models: (A)= ARIMA, (B)=FBProphet, (C)= PostMMS-LSTM4 with 5-day lookback. (D)=PostMMS-LSTM4 with no lookback, notice the lower orange line compared to subfigure(C). (E) PreMMS LSTM (F) PostMMS-LSTM-Weekly. All of them predicted a decreasing trend of incident frequency

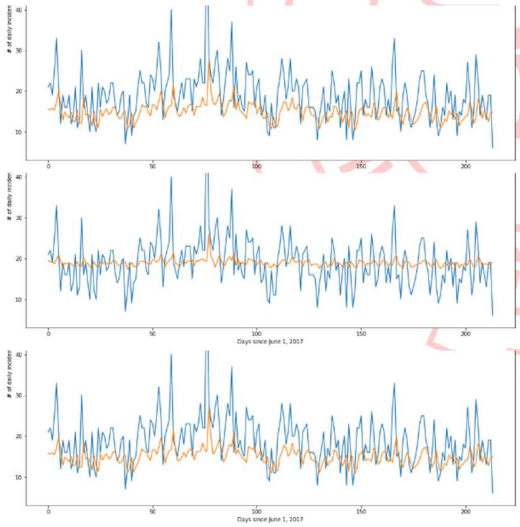


Figure 16: The PostMMS-LSTM's prediction of different configurations: (Top): Original with 4-LSTM cells, (Middle): Trained with weekly series and tested against the daily series, notice the flat graph, the worst predictions of all three. (Bottom): with 9 cells in the LSTM layer, notice the smoother troughs and higher tips. To sum up, hyperparameter tuning on simple LSTM did not evidently affect the prediction unless the training data changed.

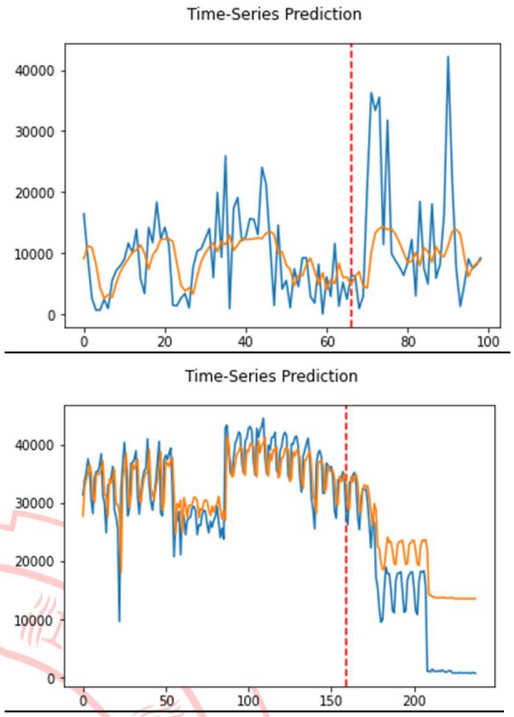


Figure 17: Pytorch-LSTM prediction results for (top) UBER and (bottom) TAXI ridership data

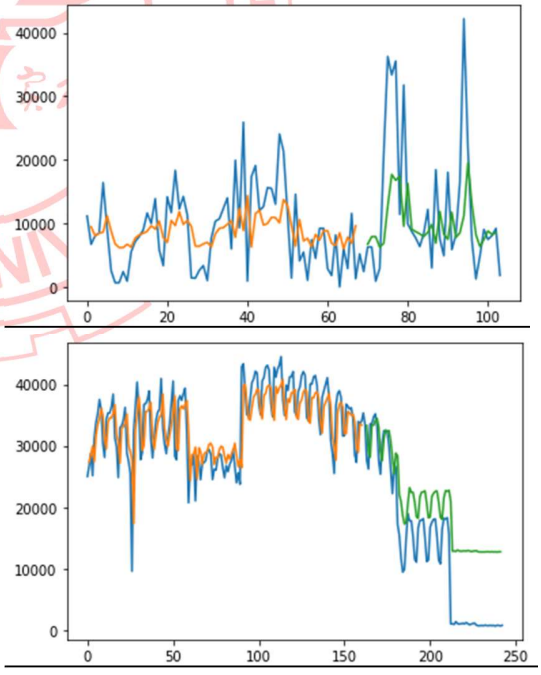


Figure 18: Keras-LSTM prediction results for (left) UBER and (right) TAXI ridership data

4.3. Accomplished Tasks

We performed the following experiment in the first episode (ECE): 1. Taxi Uber LSTMs, ST-METANET, CRANN, AGCRN, ASTGCN, DCRNN, STGRN, Deepforecast Multi-LSTM, and Spacetimeformer. With some models out of the scope and possessing incompatible metrics, the Lotto-Att-LSTM and TPA-LSTM had their results cancelled. With excessive troubleshooting and debugging, either in COLAB or TWCC, most models were able to run successfully and produced results - only two models failed to acquire their suitable environment, namely ST-METANET and Spacetimeformer.

4.4 Discussion

Restating the results in section 4.1.2, the first experiment, the ECE, arranged a competition between some notable state-of-the-art traffic forecasting models to date and discovered that, when trained with the same or similar dataset, the newer, more architecturally complex models do not always outperform the older ones. Nevertheless, the Deepforecast multi-LSTM model overtook everything on our board, proven by the lowest values of the three metrics. Besides, the ST-Metanet shares similar metrics with DRCNN-Pemsbay, STGCN, and Spacetimeformer.

For the second experiment's (the CEHE on a small NTPCfire dataset) spatial domain, the GANs, while more architecturally advanced than Autoencoders, did not outperform them (except the simplest BAE); due to GANs, compared to the AEs, requiring a large dataset, and being harder to train and hyperparameter-tune, meaning that Autoencoders suited smaller datasets (24 training images) better than GANs, because of the ease of training.

Furthermore, in the spatial CEHE apparent subtest (Table 5), adjusting batch size also affect the training state. For instance, increasing the batch size by 3 (9 -> 12) improved some models while worsening some. Thus, choosing a batch size must be based on result optimisation, which is to be done in the future. In the future, we recommend that you plan another subtest varying even more batch sizes against metrics and choose the best one producing the least errors while controlling everything else.

Usually, in the temporal domain, the LSTM greatly outperforms ARIMA but only slightly outperforms FBProphet. However, there is an exception; for time-series with no apparent seasonality, like our fire data, FBProphet outperformed the LSTM because of the FBProphet small data size requirement compared to the LSTM. Nevertheless, the PreMMS-LSTM, if well trained and tuned, could possibly defeat the FBProphet. We recommend you perform hyperparameter subtests with this model. Generally speaking, LSTM, while being a generalist, require a sizeable time series, usually with apparent seasonality, to train appropriately and

-maximise its efficiency. Evidently, FBProphet is even more generalist because, unlike the LTSM, it better suits small datasets as ARIMA does. Honestly, in the future, we would like to recommend you do even more subtests; like number of LSTM cells, the train/test ratio, etc.

Noticing the hidden hyperparameter subtest in Table 5, LSTM, a neural network, is more prone to overfitting and gradient diminish & explosion. Adding a lookback (1 -> 5), as well as adding more LSTM cells (4 -> 9), also helps with training, slightly improving metrics. Negatively, in an LSTM, extending time-series frequency hampers its learning ability. Overall, in the CEHE, the FBProphet and the VAE are our best models to be implemented on the dot-to-raster heatmap event data. The hidden subtests will eventually lead to a model design approach called "hyperparameter tuning". In the future, you may add more LSTM, AE, and GAN variants to our CEHE.

SECTION 5: CONCLUSIONS

This paper conducted a comprehensive overview of recent advances in machine and deep learning techniques for STDM. We first categorised the different data types and representations of ST data and briefly introduced the popular deep learning models used for STDM. Next, we focused on some STDM techniques using deep neural networks. Then, we reviewed recent works based on the STDM tasks, including prediction, clustering, anomaly detection, estimation and inference, and others. Next, we performed a comparative experiment of traffic prediction architectures (our ECE) to see which one produced the tiniest error in terms of MAE, MSE and RMSE, concluding that each model did well for their respective architectures and discovering two facts: One, the newer, more architecturally advanced models do not always outperform the older, simpler ones. And two, the Deepforecast Multi-LSTM [73], to date, performed the best STDM learning and prediction. For a non-graph event data prediction, we compared each popular model to generate rasterised heatmaps and perform time-series forecasting, resulting in a discovery that the VAE in the spatial domain and the FBProphet in the temporal domain did the best at their respective jobs. Also, to study how the STDM process works for a custom dataset, we chose some suitable models for each domain for generation & forecasting tasks - and compared their performance. The hyperparameter subtests in our CEHE can eventually lead to hyperparameter tuning, a model improvement measure. Finally, we listed some open problems, provided recommendations for future subtests, discussed, and pointed out the current issues, especially ones of the runtime environment, and indicated the future research directions for this fast-growing research field.

REFERENCES

[1] Doshi, J., Basu, S. and Pang, G., 2018. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*.

[2] Doshi, Jigar. 2018. Residual Inception Skip Network for Binary Segmentation. Pages 216–219 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

[3] Amit, S.N.K.B. and Aoki, Y., 2017, September. Disaster detection from aerial imagery with convolutional neural network. In 2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC) (pp. 239-245). IEEE.

[4] V. Iglovikov, S. Mushinskiy, and V. Osin, "Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition," vol. June 2017.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," ArXiv, vol. abs/2004.10934, 2020.

[6] Fang, Z., Wang, Y., Peng, L. and Hong, H., 2020. Predicting flood susceptibility using LSTM neural networks. *Journal of Hydrology*, [online] p.125734. Available at: <https://doi.org/10.1016/j.jhydrol.2020.125734> [Accessed 18 April 2021].

[7] Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

[8] S. Wang, J. Cao, and P. Yu, "Deep learning for spatio-temporal data mining: A survey," IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2020.

[9] Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic

forecasting. *arXiv preprint arXiv:1709.04875*.

[10] Correa, D., Xie, K., & Ozbay, K. (2017). Exploring the taxi and Uber demand in New York City: An empirical analysis and spatial modeling. In *96th Annual Meeting of the Transportation Research Board, Washington, DC*.

[11] Amato, F., Guignard, F., Robert, S., & Kanevski, M. (2020). A novel framework for spatio-temporal prediction of environmental data using deep learning. Scientific Reports, 2020(10), 22243. doi: 10.1038/s41598-020-79148-7

[12] Tang, Q., Yang, M., & Yang, Y. (2019). ST-LSTM: A Deep Learning Approach Combined Spatio-Temporal Features for Short-Term Forecast in Rail Transit. Journal of Advanced Transportation, 2019, Article ID 8392592. doi:https://doi.org/10.1155/2019/8392592

[13] Lu, H., Huang, D., Song, Y., Jiang, D., Zhou, T., & Qin, J. (2020). ST-TrafficNet: A Spatial-Temporal Deep Learning Network for Traffic Forecasting. Electronics, 2020(9), 1474.

[14] Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., & Zhang, J. (2019, July 25). Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/3292500.3330884

[15] de Medrano, R., & Aznarte, J. L. (2020). A spatio-temporal attention-based spot-forecasting framework for urban traffic prediction. *Applied Soft Computing*, *96*, 106615.

[16] Luo, X., Li, D., & Zhang, S. (2019). Traffic flow prediction during the holidays based on DFT and SVR. *Journal of Sensors*, *2019*.

[17] Jiawei Han, Micheline Kamber, Jian Pei, 13 - Data Mining Trends and Research Frontiers, Morgan Kaufmann, 2012, Pages 585-631.

[18] Shih, S. Y., Sun, F. K., & Lee, H. Y. (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, *108*(8), 1421-1441.

[19] Atluri, G., Karpatne, A., & Kumar, V. (2018). Spatio-temporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)*, *51*(4), 1-41.

[20] Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, *60*(1), 208-221.

[21] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

[22] Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, *14*(5), 1003-1016.

[23] Chen, X., Faghmous, J. H., Khandelwal, A., & Kumar, V. (2015, June). Clustering dynamic spatio-temporal patterns in the presence of noise and missing data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[24] Chen, M., Davis, J. M., Liu, C., Sun, Z., Zempila, M. M., & Gao, W. (2017, September). Using deep recurrent neural network for direct beam solar irradiance cloud screening. In *Remote Sensing and Modeling of Ecosystems for Sustainability XIV* (Vol. 10405, p. 1040503). International Society for Optics and Photonics.

[25] Cheng, L., Zang, H., Ding, T., Sun, R., Wang, M., Wei, Z., & Sun, G. (2018). Ensemble recurrent neural network based probabilistic wind speed forecasting approach. Energies, 11(8), 1958.

[26] Hossain, M., Rekabdar, B., Louis, S. J., & Dascalu, S. (2015, July). Forecasting the weather of Nevada: A deep learning approach. In 2015 international joint conference on neural networks (IJCNN) (pp. 1-6). IEEE.

[27] Liu, H., Mi, X., & Li, Y. (2018). Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM. Energy Conversion and Management, 159, 54-64.

[28] Rasp, S., & Lerch, S. (2018). Neural networks for postprocessing ensemble weather forecasts. Monthly Weather Review, 146(11), 3885-3900.

[29] Zaytar, M. A., & El Amrani, C. (2016). Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. International Journal of Computer Applications, 143(11), 7-11.

[30] Ke, J., Yang, H., Zheng, H., Chen, X., Jia, Y., Gong, P., & Ye, J. (2018). Hexagon-based convolutional neural network for supply-demand forecasting of ride-sourcing services. IEEE Transactions on Intelligent Transportation Systems, 20(11), 4160-4173.

[31] Lee, D., Jung, S., Cheon, Y., Kim, D., & You, S. (2018). Forecasting taxi demands with fully convolutional networks and temporal guided embedding.

[32] Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016, October). DNN-based prediction model for spatio-temporal data. In Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems (pp. 1-4).

[33] Zhu, Q., Chen, J., Zhu, L., Duan, X., & Liu, Y. (2018). Wind speed prediction with spatio–temporal correlation: A deep learning approach. Energies, 11(4), 705.

[34] Lv, J., Li, Q., Sun, Q., & Wang, X. (2018, January). T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction. In 2018 IEEE international conference on big data and smart computing (bigcomp) (pp. 82-89). IEEE.

[35] Song, X., Kanasugi, H., & Shibasaki, R. (2016, July). Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (pp. 2618-2624).

[36] Chattopadhyay, A., Hassanzadeh, P., & Pasha, S. (2018). A test case for application of convolutional neural networks to spatio-temporal climate data: Re-identifying clustered weather patterns. arXiv preprint arXiv:1811.04817.

[37] Nguyen, N. T., Wang, Y., Li, H., Liu, X., & Han, Z. (2012, December). Extracting typical users' moving patterns using deep learning. In 2012 IEEE Global Communications Conference (GLOBECOM) (pp. 5410-5414). IEEE.

[38] Zhang, J., Zheng, Y., & Qi, D. (2017, February). Deep spatio-temporal residual networks for citywide crowd flows prediction. In Thirty-first AAAI conference on artificial intelligence.

[39] Gardner Jr, E. S. (2006). Exponential smoothing: The state of the art—Part II. International journal of forecasting, 22(4), 637-666.

[40] Box, G. E., & Jenkins, G. M. (1976). Time Series Analysis. Forecasting and Control (rev. ed.).

[41] Aoki, M. (2013). State space modeling of time series. Springer Science & Business Media.

[42] Kelejian, H. H., & Prucha, I. R. (1999). A generalized moments estimator for the autoregressive parameter in a spatial model. International economic review, 40(2), 509-533.

[43] Brunsdon, C., Fotheringham, S., & Charlton, M. (1998). Geographically weighted regression. Journal of the Royal Statistical Society: Series D (The Statistician), 47(3), 431-443.

[44] Oliver, M. A., & Webster, R. (1990). Kriging: a method of interpolation for geographical information systems. International Journal of Geographical Information System, 4(3), 313-332.

[45] Kasetkasem, T., & Varshney, P. K. (2002). An image change detection algorithm based on Markov random field models. IEEE Transactions on Geoscience and Remote Sensing, 40(8), 1815-1823.

[46] Schroder, M., Rehrauer, H., Seidel, K., & Datcu, M. (1998). Spatial information retrieval from remote-sensing images. II. Gibbs-Markov random fields. IEEE Transactions on geoscience and remote sensing, 36(5), 1446-1455.

[47] Zhao, Y., Zhang, L., Li, P., & Huang, B. (2007). Classification of high spatial resolution imagery using improved Gaussian Markov random-field-based texture features. IEEE Transactions on Geoscience and Remote Sensing, 45(5), 1458-1468.

[48] Zhang, P., Jia, Y., Gao, J., Song, W., & Leung, H. (2018). Short-term rainfall forecasting using multi-layer perceptron. IEEE Transactions on Big Data, 6(1), 93-106.

[49] Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., & Li, T. (2018). Predicting citywide crowd flows using deep spatio-temporal residual networks. Artificial Intelligence, 259, 147-166.

[50] Zhao, P., Zhu, H., Liu, Y., Li, Z., Xu, J., & Sheng, V. S. (2018). Where to go next: A spatio-temporal LSTM model for next POI recommendation. arXiv preprint arXiv:1806.06671.

[51] Duan, L., Hu, T., Cheng, E., Zhu, J., & Gao, C. (2017). Deep convolutional neural networks for spatiotemporal crime prediction. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE) (pp. 61-67). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[52] Ding, D., Zhang, M., Pan, X., Wu, D., & Pu, P. (2018, April). Geographical feature extraction for entities in location-based social networks. In Proceedings of the 2018 World Wide Web Conference (pp. 833-842).

[53] Costilla-Reyes, O., Scully, P., & Ozanyan, K. B. (2017). Deep neural networks for learning spatio-temporal features from tomography sensors. IEEE Transactions on Industrial Electronics, 65(1), 645-653.

[54] Wen, D., Wei, Z., Zhou, Y., Li, G., Zhang, X., & Han, W. (2018). Deep learning methods to process fmri data and their application in the diagnosis of cognitive impairment: a brief overview and our opinion. Frontiers in neuroinformatics, 12, 23.

[55] Dvornek, N. C., Ventola, P., Pelphrey, K. A., & Duncan, J. S. (2017, September). Identifying autism from resting-state fMRI using long short-term memory networks. In International Workshop on Machine Learning in Medical Imaging (pp. 362-370). Springer, Cham.

[56] Guo, X., Dominick, K. C., Minai, A. A., Li, H., Erickson, C. A., & Lu, L. J. (2017). Diagnosing autism spectrum disorder from

brain resting-state functional connectivity patterns using a deep neural network with a novel feature selection method. Frontiers in neuroscience, 11, 460.

[57] Heinsfeld, A. S., Franco, A. R., Craddock, R. C., Buchweitz, A., & Meneguzzi, F. (2018). Identification of autism spectrum disorder using deep learning and the ABIDE dataset. NeuroImage: Clinical, 17, 16-23.

[58] Horikawa, T., & Kamitani, Y. (2017). Generic decoding of seen and imagined objects using hierarchical visual features. Nature communications, 8(1), 1-15.

[59] Kim, J., Calhoun, V. D., Shim, E., & Lee, J. H. (2016). Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: Evidence from whole-brain resting-state functional connectivity patterns of schizophrenia. Neuroimage, 124, 127-146.

[60] Meszlényi, R. J., Buza, K., & Vidnyánszky, Z. (2017). Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture. Frontiers in neuroinformatics, 11, 61.

[61] Shi, J., Zheng, X., Li, Y., Zhang, Q., & Ying, S. (2017). Multimodal neuroimaging feature learning with multimodal stacked deep polynomial networks for diagnosis of Alzheimer's disease. IEEE journal of biomedical and health informatics, 22(1), 173-183.

[62] Cheng, W., Shen, Y., Zhu, Y., & Huang, L. (2018, April). A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In Thirty-second AAAI conference on artificial intelligence.

[63] Zhang, H., Wu, H., Sun, W., & Zheng, B. (2018). Deeptravel: a neural network-based travel time estimation model with auxiliary supervision. arXiv preprint arXiv:1802.02147.

[64] Kut, A., & Birant, D. (2006). Spatio-temporal outlier detection in large databases. Journal of computing and information technology, 14(4), 291-297.

[65] Cheng, T., & Li, Z. (2004, June). A hybrid approach to detect spatial-temporal outliers. In Proceedings of the 12th International Conference on Geoinformatics Geospatial Information Research (pp. 173-178).

[66] Cheng, T., & Li, Z. (2006). A multiscale approach for spatio-temporal outlier detection. Transactions in GIS, 10(2), 253-263.

[67] Zhang, Z., He, Q., Gao, J., & Ni, M. (2018). A deep learning approach for detecting traffic accidents from social media data. Transportation research part C: emerging technologies, 86, 580-596.

[68] Zhu, L., Guo, F., Krishnan, R., & Polak, J. W. (2018, November). A deep learning approach for traffic incident detection in urban networks. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 1011-1016). IEEE.

[69] Racah, E., Beckham, C., Maharaj, T., Kahou, S. E., & Pal, C. (2016). ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. arXiv preprint arXiv:1612.02095.

[70] Wang, J., Wu, N., Zhao, W. X., Peng, F., & Lin, X. (2019, July). Empowering A* search algorithms with neural networks for personalized route recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 539-547).

[71] Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *arXiv preprint arXiv:2007.02842*.

[72] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019, July). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 922-929).

[73] Ghaderi, A., Sanandaji, B. M., & Ghaderi, F. (2017). Deep forecast: Deep learning-based spatio-temporal forecasting. *arXiv preprint arXiv:1707.08110*.

[74] F. Chollet, "Keras documentation: Variational Autoencoder," Keras, 01-Apr-2020. [Online]. Available: https://keras.io/examples/generative/. [Accessed: 19-Dec-2021].

[75] A. of the Google Dev Team, "Introduction | generative adversarial networks | google developers," Generative Adversarial Networks, 08-Oct-2019. [Online]. Available: https://developers.google.com/machine-learning/gan. [Accessed: 20-Dec-2021].

[76] M. G. Heberger, "Artificial Neural Network," Wikipedia, 02-Dec-2021. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed: 20-Dec-2021].

[77] C. Duong, "Facebook Prophet-Forecasting at scale.," Facebook Prophet, 14-Jul-2018. [Online]. Available: https://facebook.github.io/prophet/. [Accessed: 20-Dec-2021].

[78] B. Lamberta, "Github: TensorFlow Generative Models Tutorial," GitHub, 13-Mar-2021. [Online]. Available: https://github.com/tensorflow/docs/tree/master/site/en/tutorials/generative. [Accessed: 21-Dec-2021].

[79] B. Lamberta, "Deep convolutional generative Adversarial Network: TensorFlow Core," TensorFlow, 13-Mar-2021. [Online]. Available: https://www.tensorflow.org/tutorials/generative/dcgan. [Accessed: 21-Dec-2021].

[80] Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-Range Transformers for Dynamic Spatiotemporal Forecasting. arXiv preprint arXiv:2109.12218.

[81] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028.

[82] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2794-2802).

[83] S. Guo, Y. Lin, H. Wan, X. Li and G. Cong, "Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2021.3056502.

[84] J. Brownlee, "Machine learning mastery," Machine Learning Mastery, 25-Oct-2021. [Online]. Available: https://machinelearningmastery.com/. [Accessed: 13-Jan-2022].